REPORT DOCUMENTATION PAGE

1. AGENCY USE ONLY (Leave blank)	**2. REPORT DATE** 4 May 2007	**3. REPORT TYPE AND DATE COVERED**

4. TITLE AND SUBTITLE An Examination of acceptable navigation accuracy for LISA orbits **Smythe, Reid W.**	**5. FUNDING NUMBERS**
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) **US Naval Academy** **Annapolis, MD 21402**	**10. SPONSORING/MONITORING AGENCY REPORT NUMBER** **Trident Scholar project report no. 360 (2007)**

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT **This document has been approved for public release; its distribution is UNLIMITED.**	**12b. DISTRIBUTION CODE**

13. ABSTRACT

This project assessed the accuracy with which a formation of satellites, the Laser Interferometer Space Antenna (LISA), must be placed into orbit. The LISA formation will consist of three satellites orbiting the Sun, forming an equilateral triangle. The first phase dealt with the formation parameters (leg length, leg length time rate of change, interior leg angle, and formation-sun-earth angle) as a function of time. Duplication of plots contained in other papers on the topic validated the output of the analysis code. Preliminary analysis indicated the values of each parameter varied sinusoidally, and increasing the initial error conditions tended to reduce the time each parameter fell within the acceptable tolerance values. The second phase dealt with analyzing formation parameters as a function of the error size of specific initial conditions. Two of the eighteen state variables were varied simultaneously, resulting in a surface plot indicating how long a particular formation parameter was out of specification over the lifetime of the mission. The formation was most sensitive to velocity errors in the in-track direction. Furthermore, it appeared that one error can counter-act another error, agreeing with other papers on this topic. The third phase analyzed the effect of varying all eighteen variables simultaneously. A plot was created to show how the value of time out of specification for the leg length rate of change changes as the initial state variable tolerance changes. As the maximum variation in position and velocity errors increased, there was a corresponding increase in the amount of time the formation parameter was out of specification. A cross-section of the surface was created with position error range remaining constant while varying velocity error range. With an assumed maximum position error of zero, the formation tended to start going out of specification at a velocity error magnitude of ±2.6 cm/sec

14. SUBJECT TERMS LISA, satellite, formation, insertion error	**15. NUMBER OF PAGES** 60
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**

U.S.N.A. --- Trident Scholar project report; no. 360 (2007)

An Examination of Acceptable Navigation Accuracy for LISA Orbits

by

Midshipman 1/c Reid W. Smythe
United States Naval Academy
Annapolis, Maryland

____
(signature)

Certification of Advisers Approval

Visiting Professor Dr. Richard Fahey
Aerospace Engineering Department

____
(signature)

____
(date)

Professor Dr. Daryl Boden, Chair,
Aerospace Engineering Department

____
(signature)

____
(date)

Acceptance for the Trident Scholar Committee

Professor Joyce E. Shade
Deputy Director of Research & Scholarship

____
(signature)

____
(date)

Abstract

This project assessed the accuracy with which a formation of satellites, the Laser Interferometer Space Antenna (LISA), must be placed into orbit. The LISA formation will consist of three satellites orbiting the Sun, forming an equilateral triangle.

The first phase dealt with the formation parameters (leg length, leg length time rate of change, interior leg angle, and formation-sun-earth angle) as a function of time. Duplication of plots contained in other papers on the topic validated the output of the analysis code. Preliminary analysis indicated the values of each parameter varied sinusoidally, and increasing the initial error conditions tended to reduce the time each parameter fell within the acceptable tolerance values.

The second phase dealt with analyzing formation parameters as a function of the error size of specific initial conditions. Two of the eighteen state variables were varied simultaneously, resulting in a surface plot indicating how long a particular formation parameter was out of specification over the lifetime of the mission. The formation was most sensitive to velocity errors in the in-track direction. Furthermore, it appeared that one error can counter-act another error, agreeing with other papers on this topic.

The third phase analyzed the effect of varying all eighteen variables simultaneously. A plot was created to show how the value of time out of specification for the leg length rate of change changes as the initial state variable tolerance changes. As the maximum variation in position and velocity errors increased, there was a corresponding increase in the amount of time the formation parameter was out of specification. A cross-section of the surface was created with position error range remaining constant while varying velocity error range. With an assumed maximum position error of zero, the formation tended to start going out of specification at a velocity error magnitude of ±2.6 cm/sec.

KEYWORDS: LISA, satellite, formation, insertion error

Acknowledgements

First and foremost I wish to acknowledge my research advisors, Dr. Richard Fahey and Dr. Daryl Boden, who provided me with the inspiration for hours of work after short conversations. Dr. Ted Sweetser at JPL was instrumental in providing the propagator to me as well as making it work by providing his own script for LISA propagations. Dr. Stephen Merkowitz of Goddard Space Flight Center shared both his time and knowledge of LISA without which this project would have lost a lot of direction. Finally, the Trident Scholar Committee gave ample motivation and feedback to ensure that everything was finished on time.

Table of Contents

List of Figures

List of Tables

List of Symbols

$\mathbf{I}$ – Unit vector in the direction of the first axis
$\mathbf{K}$ – Unit vector in the direction of the third axis
$\mathbf{R}_{\oplus}$ – Earth position vector
$\mathbf{R}_{C}$ – Formation center position vector
$R_{\oplus}$ – Earth position vector magnitude
R_{C} – Formation center position vector magnitude
$\hat{E}$ – Unit vector in the East direction
$\hat{S}$ – Unit vector in the South direction
$\hat{Z}$ – Unit vector in the vertical direction
a – Semi-major axis
e – Eccentricity
$\dot{d}$ – Leg length rate of change
d – Leg length
$\mathbf{h}$ – Angular momentum vector
h – Angular momentum vector magnitude
i – Inclination
$\mathbf{r}$ – Position vector
r – Position vector magnitude
$\mathbf{n}$ - Line of nodes vector
n – Line of nodes vector magnitude
$\mathbf{v}$ – Velocity vector
ω – Argument of periapsis
$\mathbf{e}$ – Eccentricity vector
Az – Azimuth
el – Elevation
Ω – Longitude of ascending node
ε – Specific mechanical energy
θ – Interior leg angle
μ – Gravitational parameter
υ – True anomaly
ν_{lag} – Formation-Sun-Earth angle
ρ – Slant range

Overview

This project assessed the accuracy with which a particular constellation of satellites, the Laser Interferometer Space Antenna (LISA), must be placed into orbit in order to enter into and maintain formation over the life of the mission.

The success of this mission depends heavily on the ability to place the satellites in their proper orbits. Slight errors in the orbital insertion become larger over time, and reduce the life of the mission. Due to the current proposed construction of the spacecraft, mid-life orbit corrections are unlikely to be available, further emphasizing the necessity of the accurate placement of the satellites into the correct orbit.

By performing a detailed analysis of the effects of errors on formation parameters over time, this project can effectively tell the LISA design team the allowable tolerance in initial conditions to achieve the desired performance of the formation parameters over the life of the LISA mission.

Background

Gravitational Waves

Introductory physics courses discuss electromagnetic (EM) waves that travel in one direction with a sinusoidal electric field and magnetic field perpendicular to the direction of propagation and to each other. These are generated by the acceleration of charged particles and are the means for wireless communications. The waves create signals in antennas through the process of induction, providing an easy method of detection. Gravitational waves, on the other hand, come from the acceleration of mass. Predicted by Einstein, these waves are yet to be directly detected. In 1993, Drs. Russell Hulse and Joseph Taylor indirectly found evidence of their existence through measurements of a binary star system.[1]

Gravitational waves affect space as they travel, causing contractions and expansions. Two masses set apart at a given distance would have the distance between them changed without moving as a gravity wave passes between them. So theoretically, detection of gravitational waves would be possible if the precise distance between two masses with no relative motion between each other is continuously measured with an accurate ruler[2].

Currently there are several projects attempting to detect these waves, including the Laser Interferometer Gravitational-Wave Observatory (LIGO), TAMA (a Japanese detector), the proposed LISA project, and others. In most cases the projects use interferometers as the basis for detection.

Interferometers

Interferometers operate based on the principle of constructive and destructive

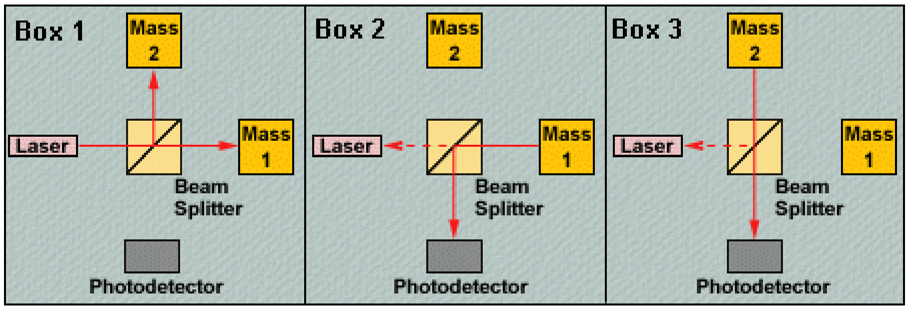


Figure 1. A Michelson Interferometer[3]

interference. Two waves in phase with each other will have constructive interference and appear stronger than either when added together, while two waves out of phase with each other will

[1] Faye Flam, “Physics: A Prize for Patient Listening.” *Science* 262 No. 5133 (1993): 507-508

have destructive interference and appear weaker than either when combined. Figure 1 provides a diagram of a typical interferometer setup. The first box shows light coming from a source and being split by a mirror into two legs. Boxes 2 and 3 show each leg being traversed and the light returning to a photo detector. When comparing the legs, constructive interference appears as bright spots while destructive interference appears as dark spots.

Given an interferometer with known distances between the legs, the interference pattern can be used to calculate the wavelength of the type of wave being examined. Likewise, a wave

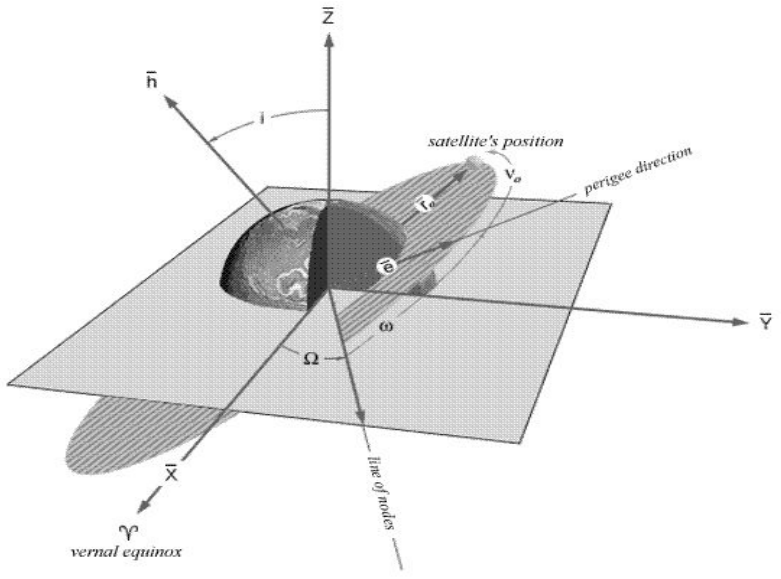


Figure 2. A Diagram of the Classical Orbital Elements[4]

of a specific wavelength can be used to measure changes in distance very precisely through Doppler shift and phase variation of the wave causing visible changes in the interference pattern.

---

[2] Peter Saulson, "If light waves are stretched by gravitational waves, how can we use light as a ruler to detect gravitational waves?" American Journal of Physics 65, no. 6 (1997): 501-505.
[3] "How LISA Works" n.d., <http://lisa.nasa.gov/TECHNOLOGY/LISA_interfer.html> (19APR2007)
[4] Kim Dismukes, "Orbital Elements." 2002. <http://spaceflight.nasa.gov/realdata/elements/graphs.html> (19APR2007)

Satellites

The first man-made object to orbit the Earth, *Sputnik*, signaled the beginning of serious space operations. Many types of satellites were quickly developed including telecommunications, remote sensing, and exploration.

Table 1. Equations for the Classical Orbital Elements[5]

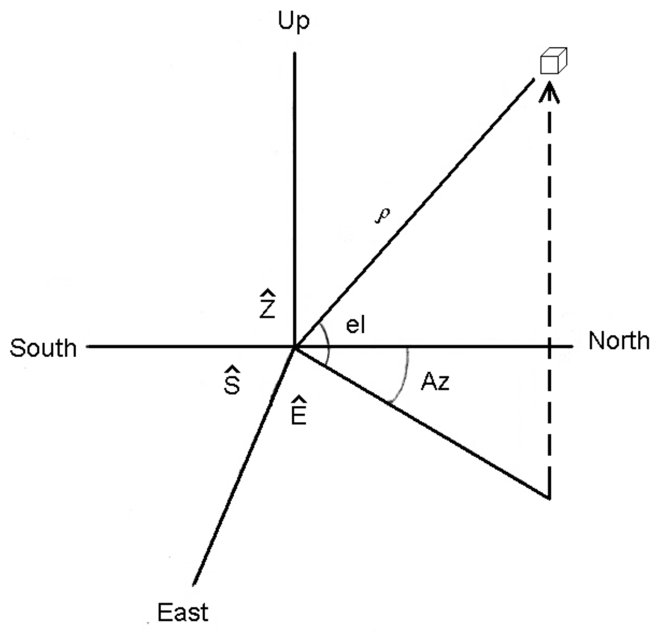
Orbital Element	Equation
Semi-major Axis	$a = \frac{-\mu}{2\varepsilon}$
Eccentricity	$e = \left\lvert \frac{1}{\mu}\left(\left(\lVert\mathbf{v}\rVert^2 - \frac{\mu}{r}\right)\mathbf{r} - (\mathbf{r}\cdot\mathbf{v})\,\mathbf{v}\right)\right\rvert$
Inclination	$i = \cos^{-1}\left(\frac{\mathbf{h}\cdot\mathbf{K}}{h}\right)$
Longitude of Ascending Node	$\Omega = \cos^{-1}\left(\frac{\mathbf{n}\cdot\mathbf{I}}{n}\right)$
Argument of Periapsis	$\omega = \cos^{-1}\left(\frac{\mathbf{n}\cdot\mathbf{e}}{ne}\right)$
True Anomaly	$\nu = \cos^{-1}\left(\frac{\mathbf{e}\cdot\mathbf{r}}{er}\right)$

All types of satellites orbit in the same way: they fall towards the object about which they orbit, and miss it; they are in constant motion. This motion can be described on a basic level by using Kepler's laws: the Earth orbits the Sun in an ellipse with the Sun at one focus, a line connecting the Earth and the Sun will sweep out equal areas in equal time intervals, and the square of the orbital period is proportional to the cube of the major axis of the orbit.

Six elements are needed to determine particular orbits, as seen in Figure 2, with associated equations in Table 1. These Classical Orbital Elements (COEs) describe the basic components of the orbit, where is a constant called the gravitational parameter, and ε is the specific mechanical energy of the satellite. The semi-major axis defines the size of the orbit, the eccentricity defines how circular the orbit is, and the inclination defines how the orbit is oriented with relation to the body being orbited. The longitude of ascending node indicates where the orbit crosses the equatorial plane of the object being orbited, the argument of periapsis is the closest point between the orbit and the object being orbited, and the true anomaly defines where the satellite currently is along the orbit.

There are special cases of orbits where one or more of the terms must be changed (such as circular orbits), however none of those apply to this project. These orbital elements can be calculated from position vector and velocity vector data (**r** and **v**), which can be determined through the analysis of azimuth, elevation, range, and the time derivative of each. The azimuth, elevation, range and rate of change of each come from a tracking system. They are given in the topocentric (location centered) horizon coordinate system as seen in Figure 3.

Actual orbits vary with time. Environmental perturbations cause these variations. For



objects in Earth orbit, perturbations include such things as atmospheric drag, irregular gravity fields due to a non-spherical, non-homogeneous Earth, out-gassing of the spacecraft, solar wind,

[5] David Vallado, Fundamentals of Astrodynamics and Applications. New York: McGraw-Hill, (1997): 130.

solar radiation, the influence of third bodies such as Jupiter, and many others. When determining where a satellite will be at a given time, all factors must be included in the calculations or the answer will not be accurate. Furthermore, measurement instruments all have inherent errors. Radar must travel through the atmosphere which can affect the wave's velocity and path. There is a measurable time delay between when the radar wave is transmitted and received. The radar equipment may be slightly mis-calibrated. Thus, orbits always change slightly, and there is some amount of error present in all operations. Even planning orbits introduces some amount of uncertainty, as the propagator, the software package used to predict the orbit, may not include nor model in the same manner all perturbations that will affect the system, and the finite precision of computers produces some rounding errors.

Orbit Propagation and Analysis

Orbit propagation involves the prediction of where a satellite will be at a future point in time given a set of current conditions. These conditions can come in a variety of ways: orbital elements, position and velocity vectors, or azimuth, elevation, range and associated time rates of change.

A propagator takes the differential equation of motion for a two-body system given by Equation (1), and integrates it over some specified period of time. In an ideal case, this would mean that only true anomaly, ν, changes.

$$\ddot{\mathbf{r}} + \frac{\mu}{r^3}\mathbf{r} = \mathbf{0} \tag{1}$$

Thus, given a new value for ν the old values for the rest of the COEs can be used to calculate new **r** and **v**. Unfortunately, perturbations cause other orbital elements to vary with time.

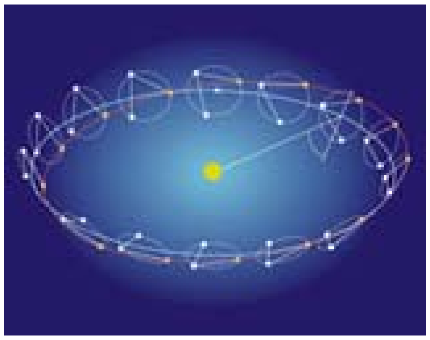
Propagators used to design actual orbits take into account perturbations when calculating a position based on data, and the accuracy depends on how precisely the perturbations have been implemented in the program. This means that the accuracy of the propagator results will decrease as it calculates further and further into the future.

So long as unaccounted external forces do not interact with objects in orbit, variations in one component of the position and velocity vectors will have a calculable affect on the other components. For example, assume a satellite has a known position <x, y, z> and velocity <u, v, w>. This information allows an orbit to be predicted. If a micro-meteorite hits the satellite and changes the value of u slightly, all other later values for x, y, z, v, and w will be different than what the initial orbit design would predict.

By varying the initial state of every variable and propagating over the design life of a mission, the effect of insertion errors on an orbit over time can be calculated. From this, the insertion accuracy can be determined based on the required tolerance of the orbit. Analyzing a large number of simulated orbits whose parameters are bounded by known error is termed a Monte-Carlo method: the use of a sufficiently large base of statistical data on a system allows the system to be numerically modeled. This is a much simpler method for modeling a complex system than analytically deriving exact solutions.

Analysis

The LISA project will place three satellites in orbit around the Sun as seen in Figure 4. The formation will trail the Earth by about 20° as measured by the formation-sun-earth angle,



t[6]

and orbit in the shape of an equilateral triangle with leg lengths of five million kilometers. Within each satellite will be a test mass free of external forces. Each leg will be connected by a laser, forming a Michelson interferometer shown in Figure 5. As gravitational waves pass through the legs of the interferometer, the distance between the test masses will change. This will be detected by combining the signals from the two legs and examining the interference pattern due to the Doppler shift.

Table 2. LISA Formation Parameter Limits

Formation Parameter	Specification
d (Leg Length)	$5 \cdot 10^6 \pm 1 \cdot 10^5$ km
$\dot{d}$ (Leg Length Time Rate of Change)	0±15 m/s
θ (Interior Leg Angle)	60°±1.5
v_{lag} (Formation, Sun, Earth Angle)	22.5°±2.5

Due to the dynamic nature of the orbits, there will be some natural oscillation of the leg-lengths and angles between the satellites. Figure 6 depicts the formation parameters which have

[6] "Mission Strategy", n.d., <http://lisa.jpl.nasa.gov/STRATEGY/getThere.html> (19APR2007)

strict requirements regarding these oscillations: the rate of change for each leg length ($\dot{d}$) can not exceed 15 m/s, the leg length (d) must be within 2% of 5 million km, the interior angles (θ)

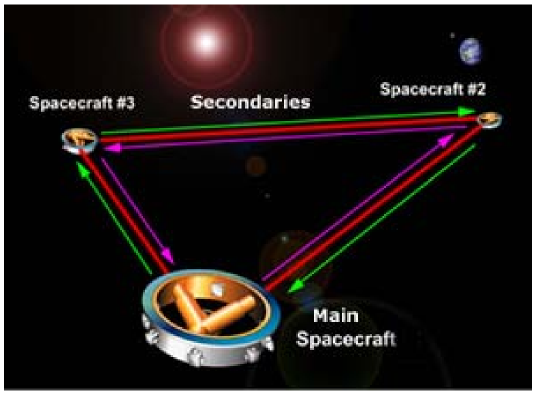


Figure 5. The LISA Formation as an Interferometer[8]

must be 60°±1.5, and the lag angle between the formation and the Earth in the Earth's orbit (ν_{lag}) can be no less than 20° and no more than 25°[7]. See Table 2.

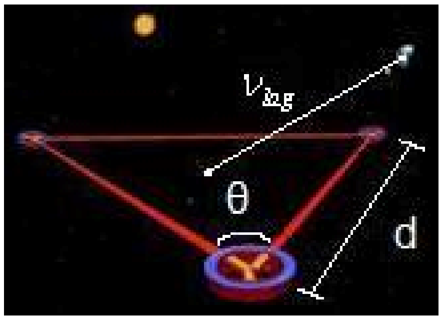


Figure 6. The LISA Formation Parameters[9]

The propagator used for this project is LTool. Developed for JPL, it is a library of Python scripts that takes into accounts such things as solar radiation, the gravitational effects of Jupiter, and other perturbations when propagating orbits. Python itself is a modern, platform-

[7] Hughes, Steven, "Preliminary Optimal Orbit Design for the Laser Interferometer Space Antenna (LISA)."

independent language that has been in use since the early 1990s. Since LTool has already been used for the mission planning of previous JPL missions, the software can be considered mature and able to produce reliable data.

Phase One - Insertion Error Analysis

Background

One way to minimize the unwanted formation changes occurs at orbital insertion. Achievement of the design orbit occurs by entering the orbit as closely as possible to the desired

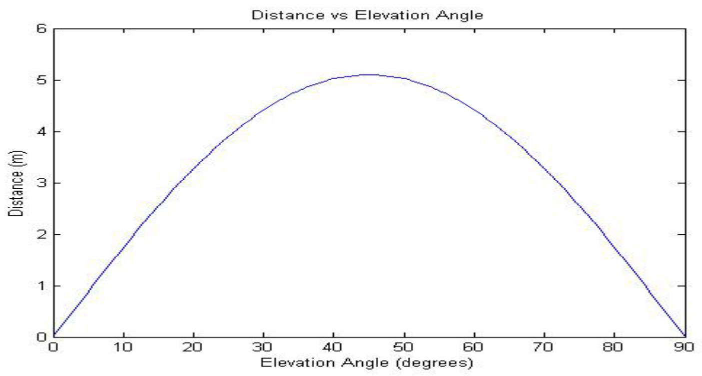


Figure 7. A Simple Ballistics Example

point with the desired velocity at the desired time. This depends on the navigation system of the craft used to insert the satellites. With six variables describing the state of each satellite at insertion (epoch), there are a total of eighteen input variables that must be varied to properly examine the system. To put this in perspective, consider a simple ballistics problem: if a gun fires a projectile at some unknown angle with a given initial velocity, how far will the projectile travel? This problem contains one single variable, the angle at which the gun fires. Thus, a

[8] "LISA Interferometry" n.d., <http://lisa.jpl.nasa.gov/TECHNOLOGY/LISA_interfer2.html> (19APR2007)
[9] "How LISA Works" n.d., <http://lisa.jpl.nasa.gov/TECHNOLOGY/challenges.html> (19APR2007)

model can be easily constructed and represented on a two-dimensional plot as seen in Figure 7. If the initial velocity also varies, then the problem becomes more complex. Figure 8 shows the solution to the two variable ballistics problem, with the colors indicating height along the vertical axis. Beyond two variables, a simple input to output relationship becomes difficult to picture.

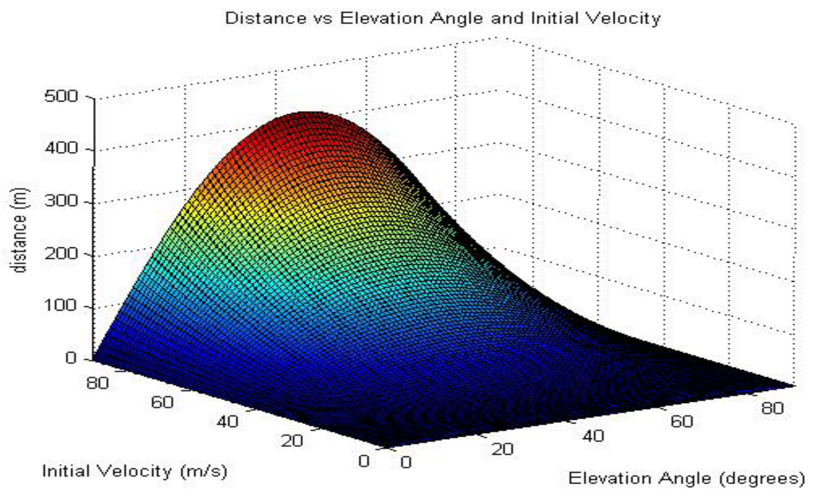


Figure 8. A Complex Ballistics Example

Formation Analysis With Respect to Time

The first step in this analysis involved developing the necessary algorithms to convert the output of the propagator into the formation parameters. To achieve this the data are imported into MATLAB®, and the position and velocity vectors are manipulated via Equations (2), (3), (4), and (5).

$$d_{i,j} = \left|\mathbf{R}_i - \mathbf{R}_j\right| \tag{2}$$

$$\dot{d}_{i,j} = \left(\mathbf{V}_i - \mathbf{V}_j\right)\cdot\left(\mathbf{R}_i - \mathbf{R}_j\right) \tag{3}$$

$$\theta_{j,i,k} = \cos^{-1}\left(\frac{\left(\mathbf{R}_i - \mathbf{R}_j\right)\cdot\left(\mathbf{R}_i - \mathbf{R}_k\right)}{d_{i,j}d_{i,k}}\right) \tag{4}$$

$$\nu_{lag} = \cos^{-1}\left(\frac{\mathbf{R}_\oplus \cdot \mathbf{R}_C}{R_\oplus R_C}\right) \tag{5}$$

Each of these parameters is then plotted with respect to time. In the case of ideal initial conditions, Figure 9 shows $\dot{d}_{1,2}$ over the course of ten years. The data for the initial condition

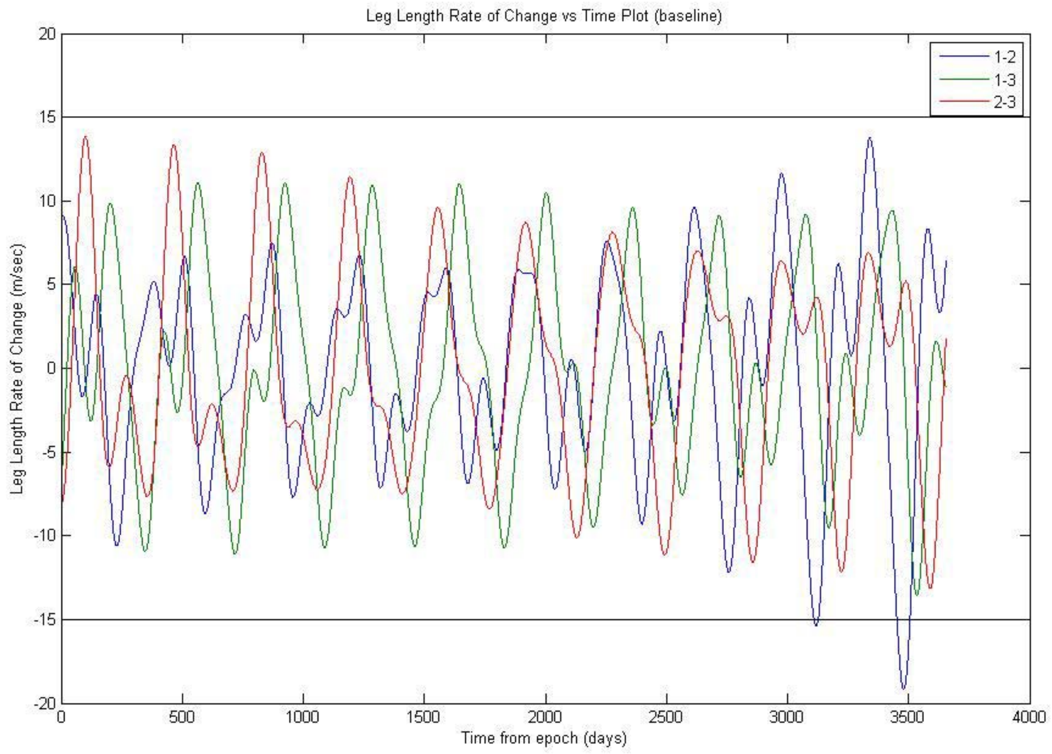


Figure 9. Baseline Leg Length Rate of Change for LISA

were validated at this point in that the graph produced from this analysis matched the graph produced in published papers on the topic of LISA orbits[10]. Following this successful analysis,

[10] Ted Sweetser, "LISA Mission Description Version 2.1." 2005.

the next step involved introducing errors into the nominal conditions and seeing how they affected the parameters.

To make the errors more understandable with respect to their orientation, all vectors were converted to the RTN coordinate system shown in Figure 10. The radial axis (R) goes in the direction from the central orbited body to the spacecraft, the cross-track axis (N) is in a direction

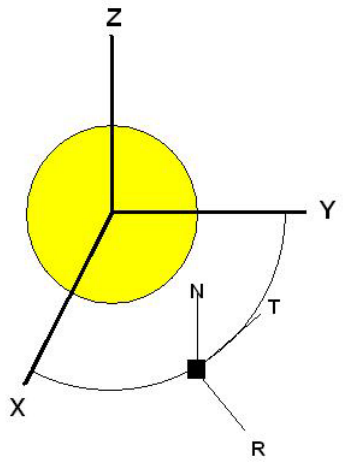


Figure 10. The RTN Coordinate System

perpendicular to the ecliptic plane, and the in-track direction (T) is perpendicular to the plane formed by the radial and cross-track directions axes. This allowed simple understanding of how the initial errors affected the initial spacecraft positions and velocities. An arbitrary error composed of $8 \cdot 10^5$ meters in the radial direction and -2 m/sec in the radial velocity was inserted into the initial conditions of spacecraft one and propagated over the same ten year period. Figure 11 shows the output for $\dot{d}_{1,2}$. $\dot{d}_{2,3}$ remains unaffected from the change, while both $\dot{d}_{1,2}$ and $\dot{d}_{1,3}$ show a less than ideal curve with both exceeding the limit of 15 m/s for a significant amount of time. Both results were as expected, however this approach did not represent a good way to examine many variations of the initial conditions since only one error condition could be analyzed at a time.

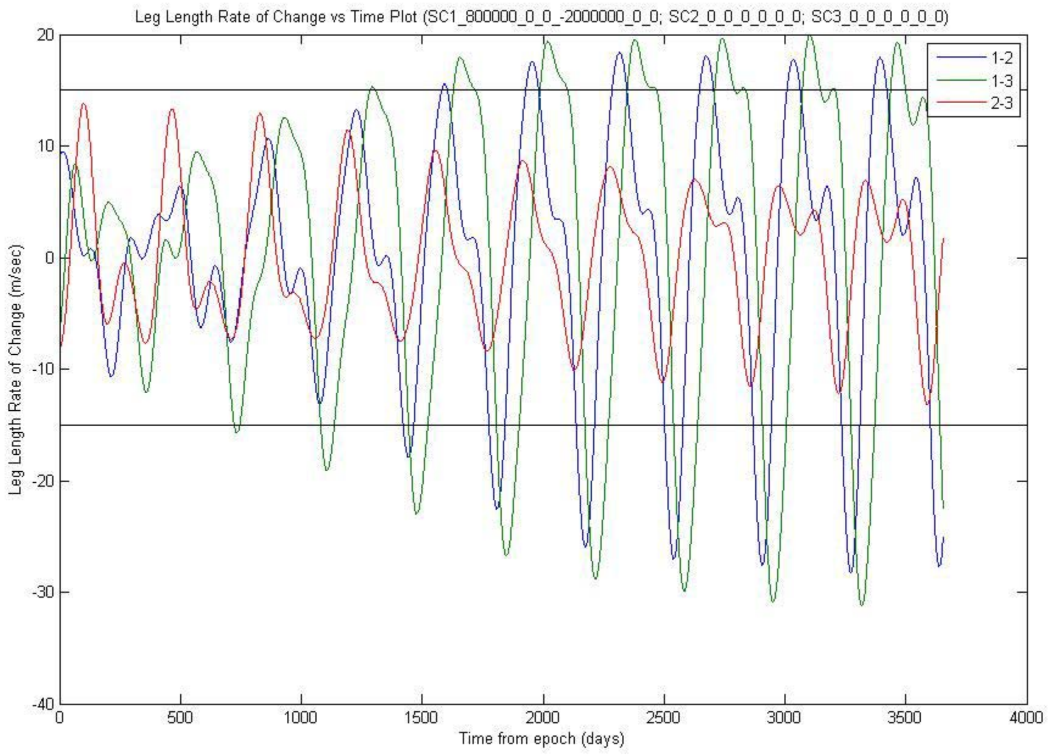
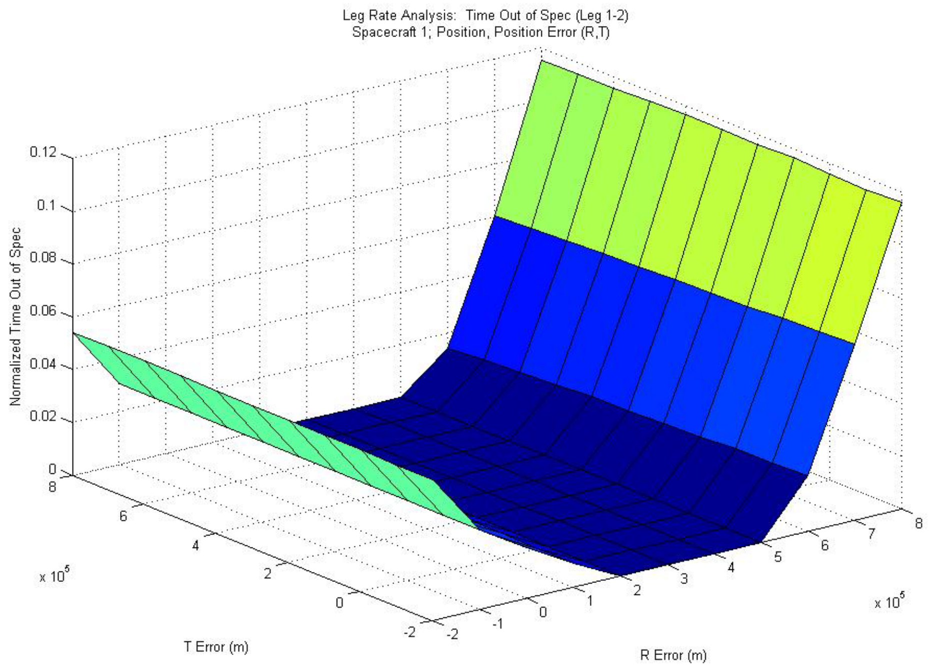


Figure 11. Leg Length Rate of Change Plot with Inserted Error

Formation Analysis With Respect to Two Input Variation

Moving from the time domain to an initial position and velocity variation domain required the creation of a metric by which each formation could be analyzed and compared to the others. Following discussions with Dr. Stephen Merkowitz at Goddard Space Flight Center, it was determined that percent time out of specification for the leg length rate of change parameter for all legs would be the best metric. Due to the sinusoidal behavior of the formation parameters as seen in Figure 11, the resulting analysis could be easily compressed and stored using only the times of the specification-boundary crossings.

Since eighteen initial parameters exist, and this analysis takes into account only two, there are several different sub-types to consider. Formation sensitivity to position or velocity errors along different directions forms one subtype, as seen in Figure 12. This compares how



either a position or velocity error along one direction affects the metric versus a similar error along a different direction, with the colors again denoting height along the vertical axis. Figure 13 shows sensitivity to position error versus sensitivity to velocity error in specific directions. This allows comparisons between initial position and initial velocity errors, including how having them occur in different directions changes the formation's sensitivity to each. Finally, the analysis can include an examination of how the position or velocity error of two different spacecraft affects the metric, shown in Figure 14. In Figures 12-14, I assumed the other 16 variables remained at their nominal values. Appendix A contains an example of a partial analysis of the Leg Length Rate of Change parameter for leg 1-2.

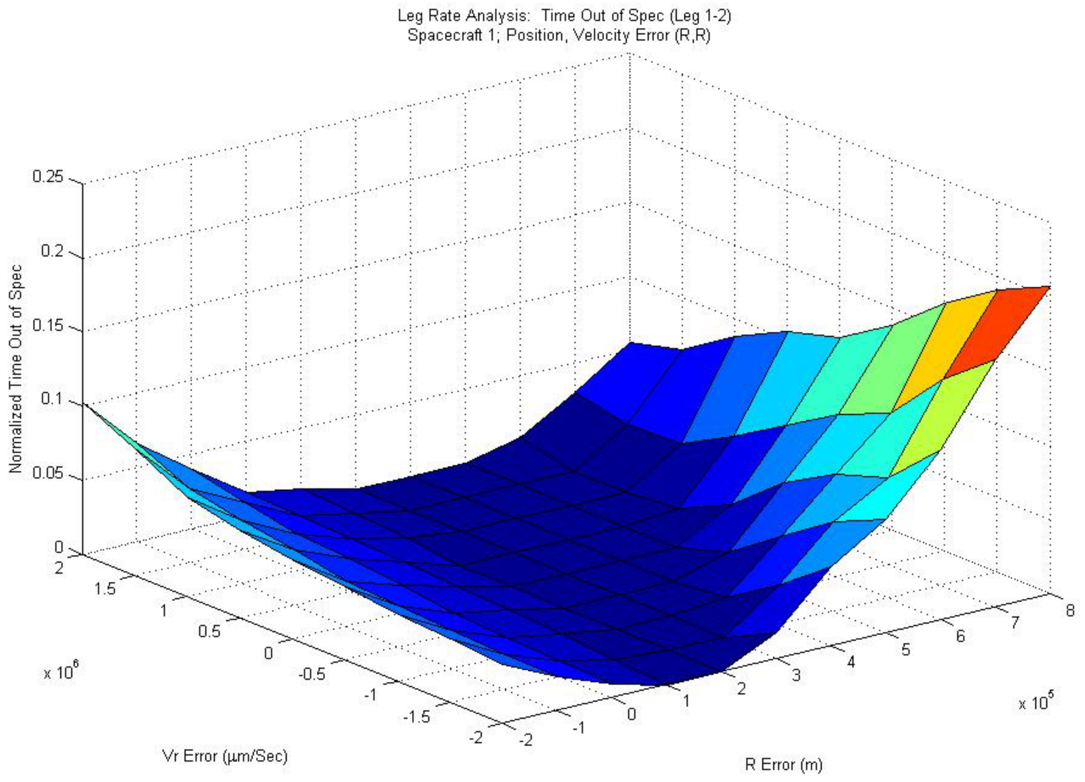


Figure 13. R-V_R Error Analysis

The shape of each of the previous three figures indicates that all initial parameters have an ideal value that minimizes the time out of specification, and values above or below that adversely affect the formation. Furthermore, the slope of the surface at a given point represents the sensitivity of the particular formation parameter being viewed to changes in initial condition. It also appears that errors can essentially cancel each other out in terms of their affect on the leg length rate of change metric.

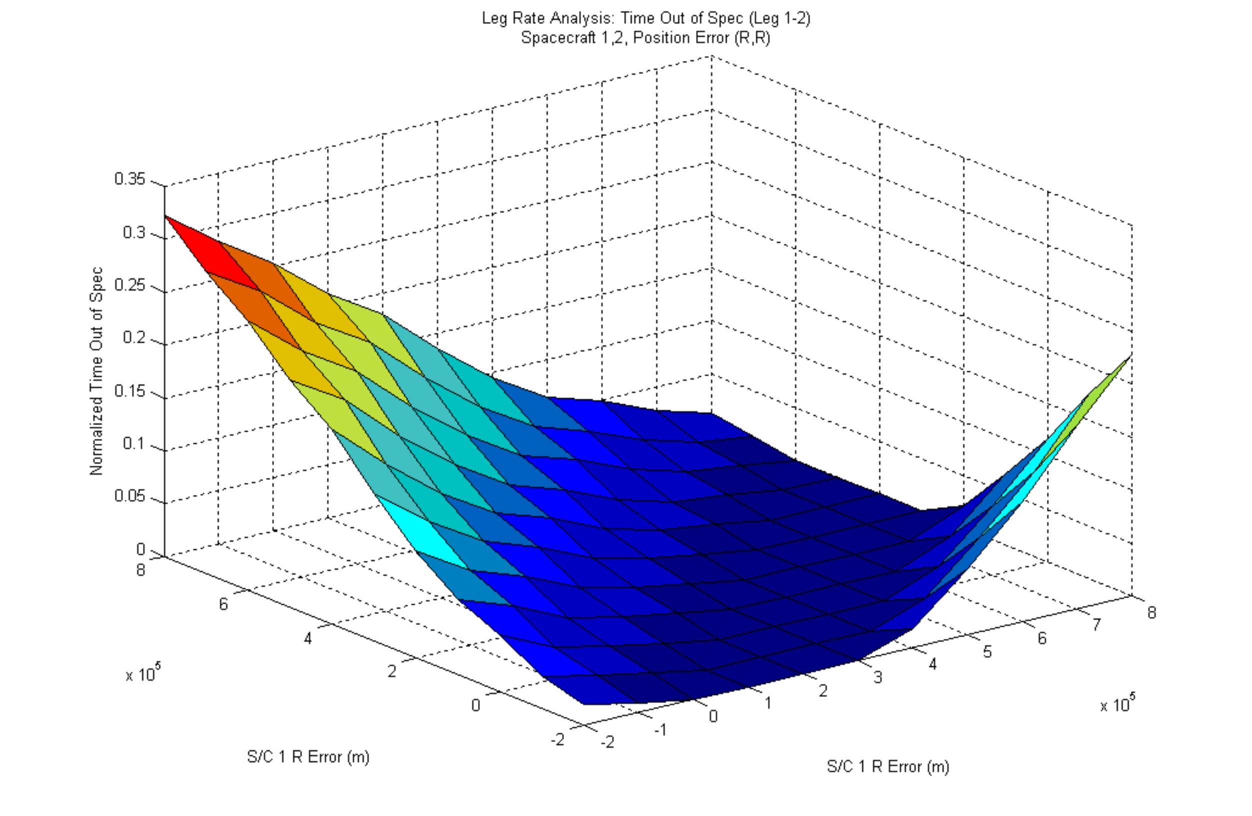


Figure 14. $R_{S/C\ 1}$ (right axis) vs $R_{S/C}$ 2 (left axis) Analysis

Formation Analysis of Full Input Variation

Because the propulsion vehicle will introduce error in all directions for both position and velocity, the analysis of two-input variation does not accurately represent a real situation. However, it is very difficult to visualize 18-dimensional space. This led to the creation of a new approach to represent the effects of errors on the formation metric: the "rainplot". Achieving the level of detail seen in the two-parameter variation analysis did not seem realistic for 18 dimensions. A reduction in the information presented by the input error axes to only the maximum variation of the position and velocity variables provided a plane on which the formation metric along the orthogonal axis could be plotted.

A set of initial formation conditions based on evenly distributed errors whose magnitude fell within the maximum variation given by a coordinate on the input Position-Velocity (P-V) plane provided the orbital conditions to propagate and analyze. The output consisted of plotting the time out of specification metric for the leg length time rate of change formation parameter along the perpendicular axis directly above the P-V coordinate for that run.

By assuming that the metric would converge around a number, given a large enough sample size, the data from many different runs could be represented by single points. To limit the time required to create each point 50 runs was used as the standard sample size. Clumping of these points around a particular value indicates the validity of the convergence assumption. By testing different coordinates on the P-V plane, the plot shows the overall formation sensitivity to

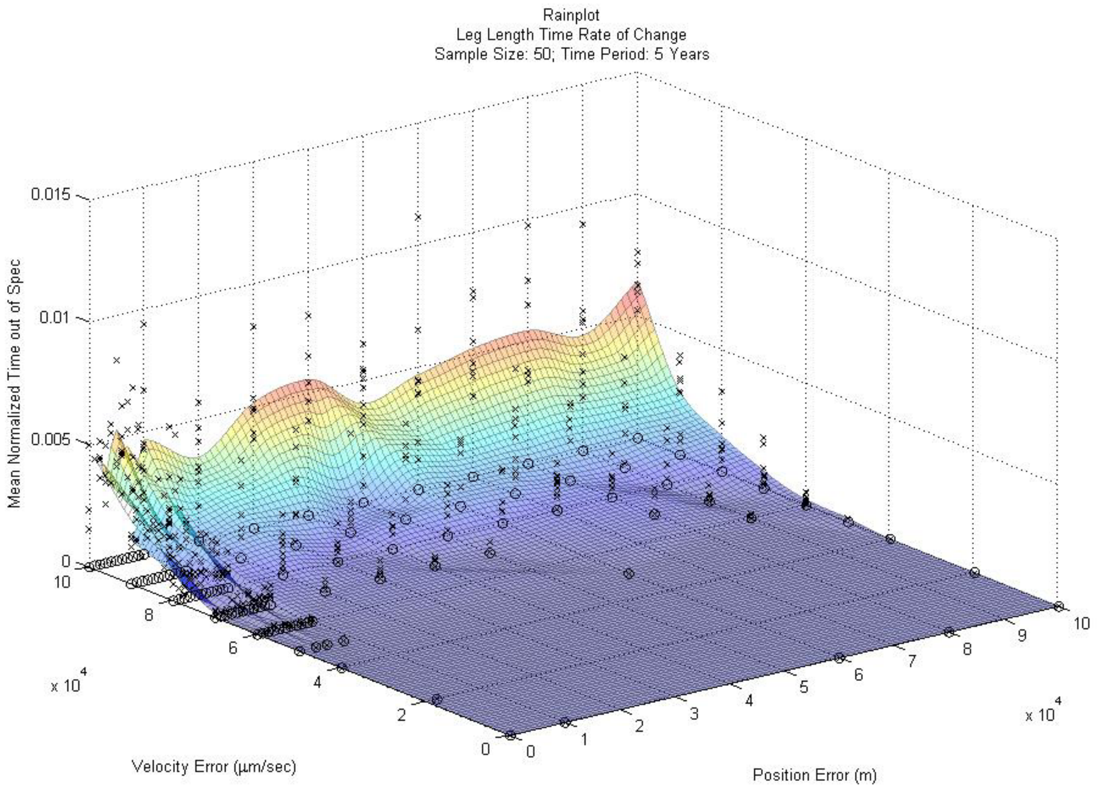


Figure 15. The Rainplot

bounded uncertainties. Figure 15 shows the results after roughly 700 runs and the analysis of over 105,000 propagations. Each 'x' represents one run, while an 'o' is plotted on the P-V plane directly under each 'x' to more easily identify the error coordinates.

The expected trend of increased mean percent time out of specification as the error limits are increased in both directions is clearly visible, with a much greater sensitivity to velocity errors rather than position errors on the scale presented. The peaks and valleys associated with the rainplot indicate that statistical convergence did not occur.

Since the velocity error dominates again, creating a cross-section of the rainplot at some specific position error provided a less computationally intensive method to obtain similar results. Figure 16 shows a cross-section where position error is assumed to be zero. This plot is

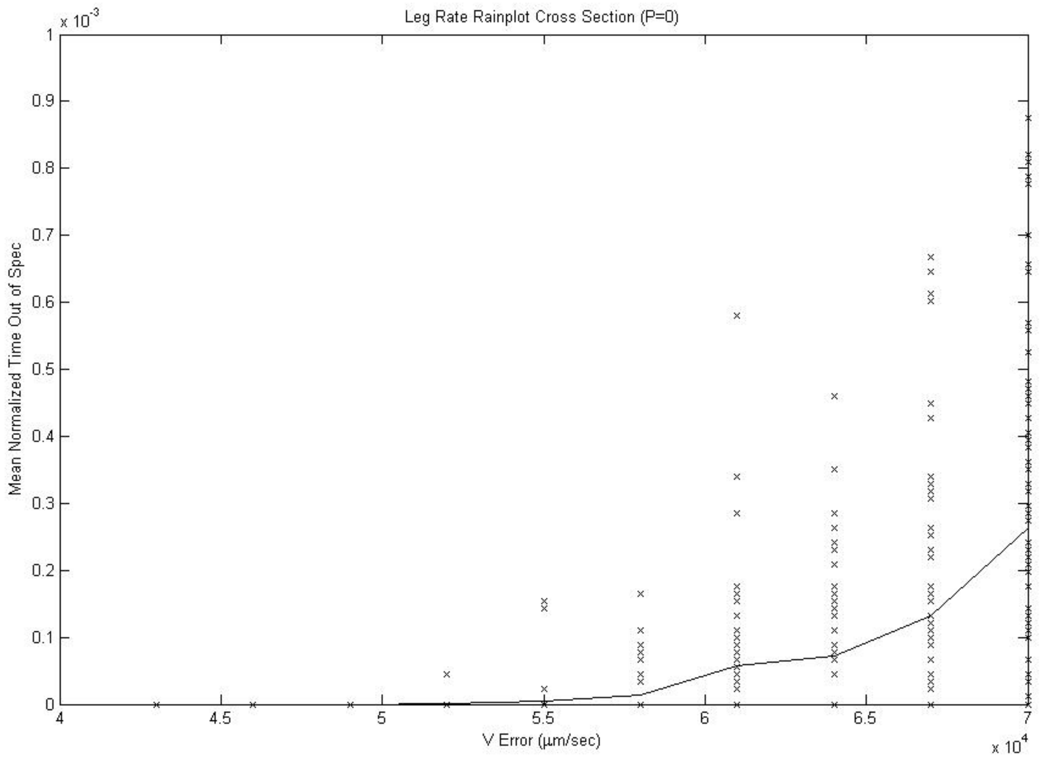


Figure 16. The Rainplot Cross-section

composed of 720 runs spread through 10 different velocity error values with a total of over 108,000 propagations analyzed. The line on the plot connects each the mean of each tested value and is not a regression. The data clearly have started to converge to what appears to be a power function, with one outlier at V = 6.1 cm/s. Though most likely a statistical aberration, it could be due to resonance between the MATLAB® random number generator and the propagator. Less likely, there could be some astrodynamical cause.

Conclusions

Through the use of increasingly complex methods of analysis, the dynamics of the formation of the three LISA satellites can be effectively examined in an error domain. It has been shown that position and velocity errors have a preferred direction along which they have the greatest effect on the formation. In particular, the formation appears to be much more sensitive to velocity errors in the in-track direction rather than position errors in the radial direction. It also appears that for the examined metric, leg length rate of change, insertion errors can cancel their effects.

Both the rainplot and the two input variation analyses indicate that the velocity error terms dominate the dynamics of the formation with regards to leg length rate of change. Because of this, a cross-section of the rainplot surface can be examined in greater detail for the same computational requirements to examine a specific range of velocity error for a given position error. As the error threshold increases, the expected time out of specification also goes up. Furthermore, it appears that until the error threshold in the velocity direction reaches ±2.6 cm/sec the formation will tend to stay in specification for the leg length rate of change parameter.

Future Work

Phase Two - Tracking System Accuracy

Beyond insertion errors, tracking errors affect how the formation is viewed. This information is important so the natural dynamics of the formation as each satellite goes around its orbit causes the formation to change. This must be accounted for so that the change in length due to the gravitational waves can be observed. Tracking will occur through NASA's Deep Space Network.

The Deep Space Network is composed of three communications facilities spaced approximately 120 degrees of longitude apart with one in Spain, one in California, and one in Australia.[11] This network provides the capability to track and communicate with spacecraft to a very great distance, or with a weak communications system. It can track objects very accurately; however, as with any real system, it has inherent error due to noise, design tolerances, and similar causes. There are thus two groups of errors: precision errors caused by the finite number of decimal places available to the computers, and accuracy errors due to physical measurement.

These inherent errors cause the satellites to be observed with different state vectors than those they actually have, which changes the observed formation. This can affect the ability of the LISA team to successfully perform the mission and detect gravitational waves. A follow-on stude of these errors would be the next step in this process.

[11] "Deep Space Network Home Page" n.d., <http://deepspace.jpl.nasa.gov/dsn/> (19APR2007)

References

Deep Space Network Home Page. n.d., <http://deepspace.jpl.nasa.gov/dsn/> (07DEC06).

Dismukes, Kim, "Orbital Elements." 2002. <http://spaceflight.nasa.gov/realdata/elements/graphs.html> (19APR2007).

Flam, Faye, "Physics: A Prize for Patient Listening." Science 262 no. 5133 (1993).

"How LISA Works" Laser Interferometer Space Antenna. n.d., <http://lisa.jpl.nasa.gov/TECHNOLOGY/challenges.html> (14FEB06).

"How LISA Works" Laser Interferometer Space Antenna. n.d., <http://lisa.nasa.gov/TECHNOLOGY/LISA_interfer.html> (19APR2007).

Hughes, Steven, "Preliminary Optimal Orbit Design for the Laser Interferometer Space Antenna (LISA)."

"LISA Interferometry" Laser Interferometer Space Antenna. n.d., <http://lisa.jpl.nasa.gov/TECHNOLOGY/LISA_interfer2.html> (19APR2007).

"Mission Strategy" Laser Interferometer Space Antenna. n.d., <http://lisa.jpl.nasa.gov/STRATEGY/getThere.html> (19APR2007)

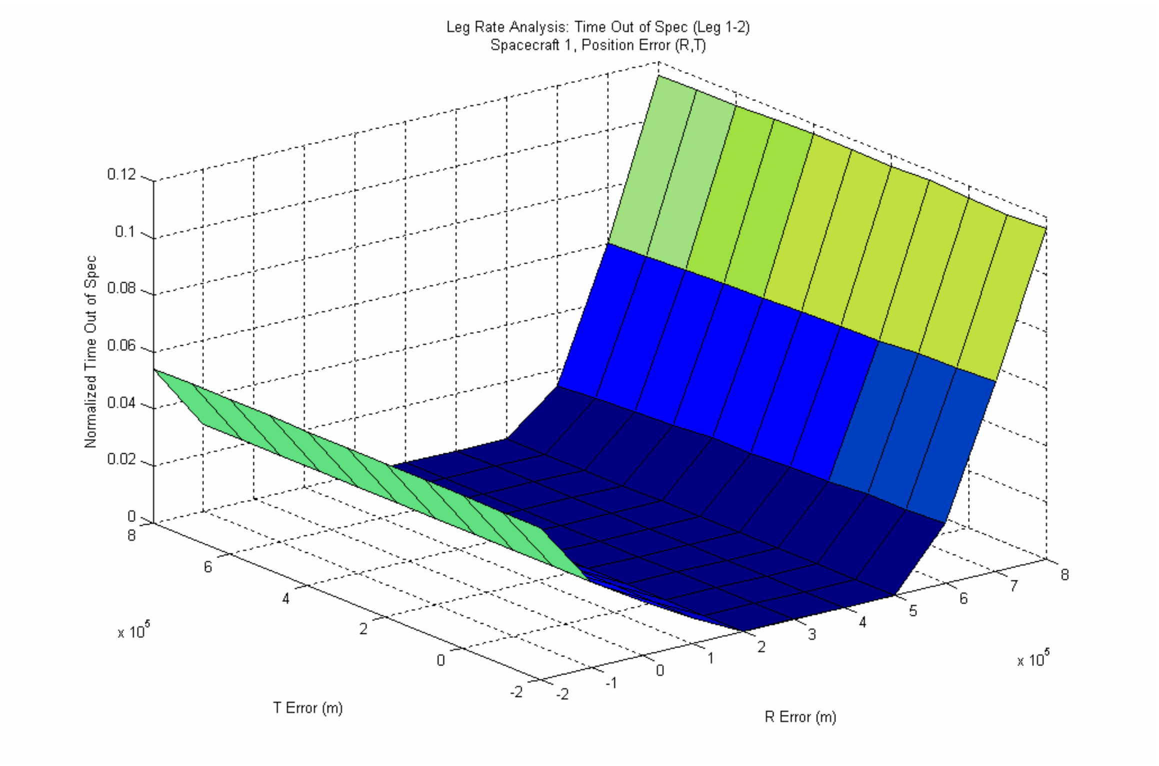
Saulson, Peter. "If light waves are stretched by gravitational waves, how can we use light as a ruler to detect gravitational waves?" American Journal of Physics 65, no. 6 (1997).

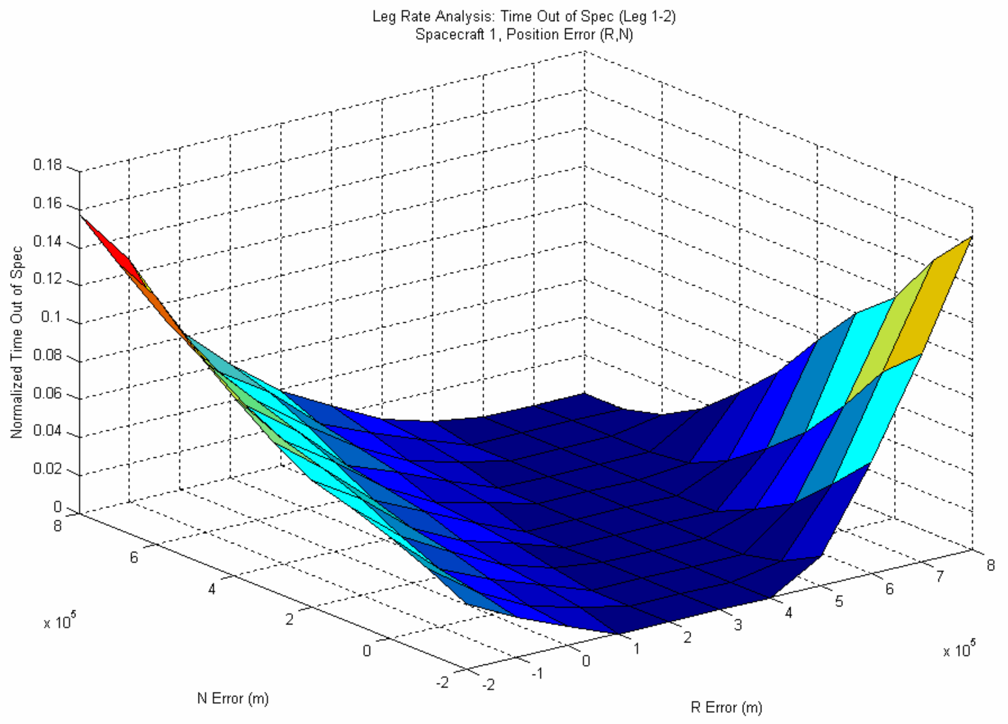
Sweetser, Ted. "LISA Mission Description Version 2.1." 2005.

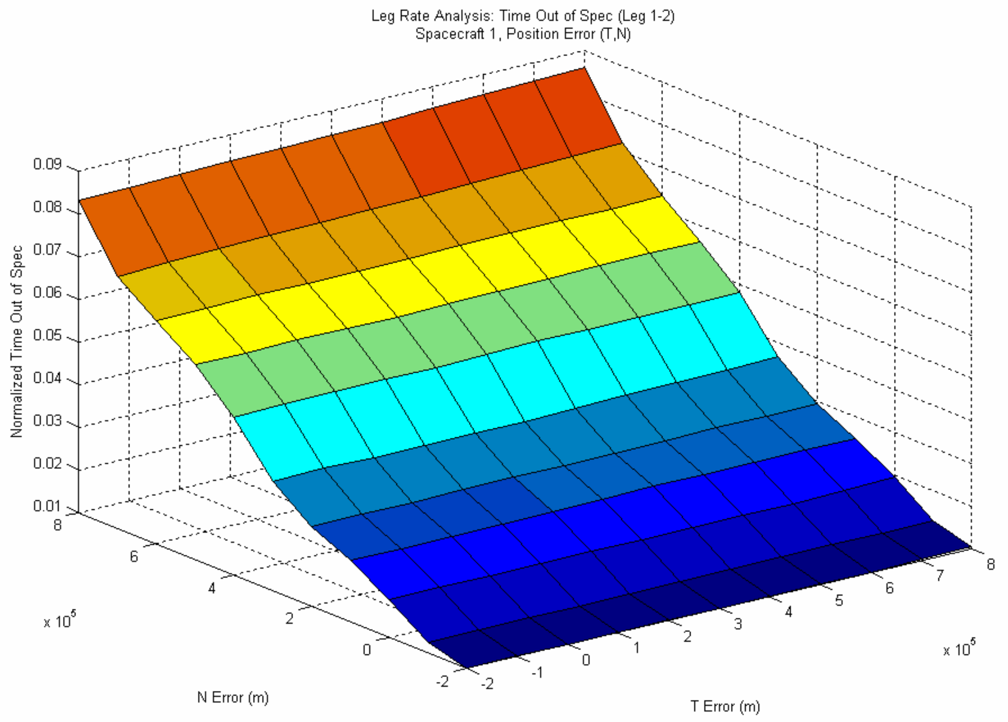
Vallado, David. Fundamentals of Astrodynamics and Applications. New York: McGraw-Hill, 1997.

Appendix A – Sample Analysis

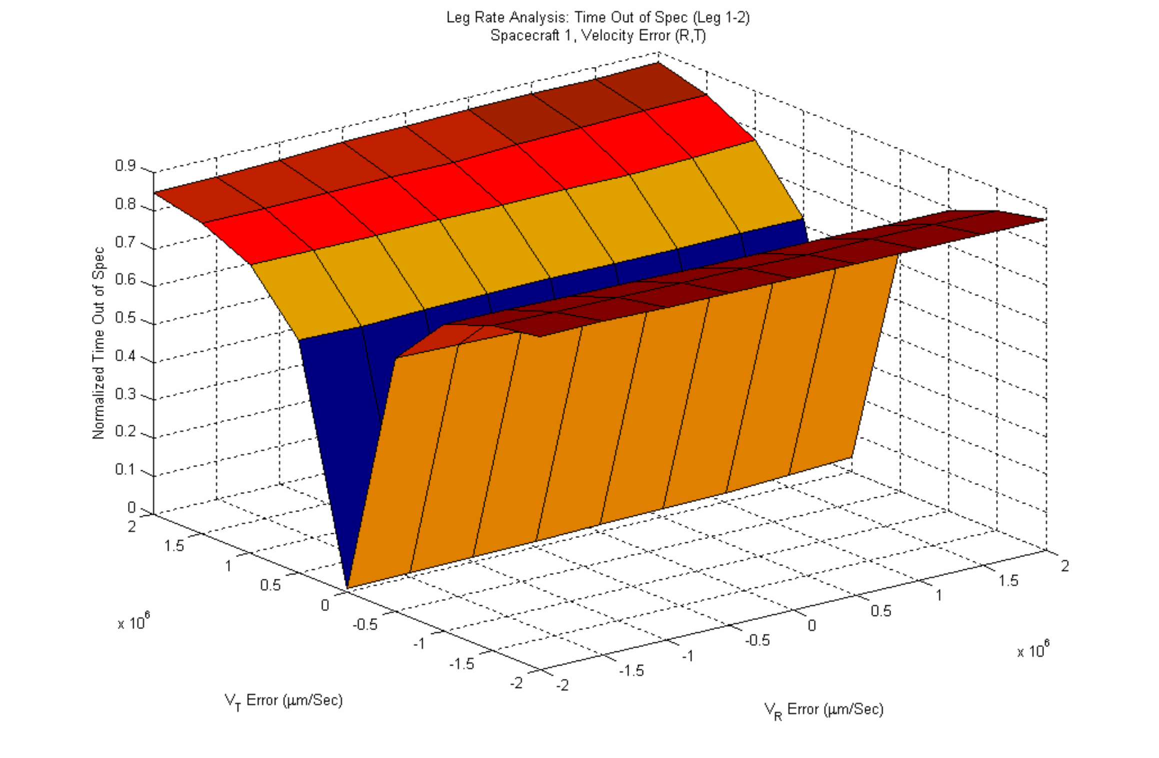
Position Errors Along Various Directions

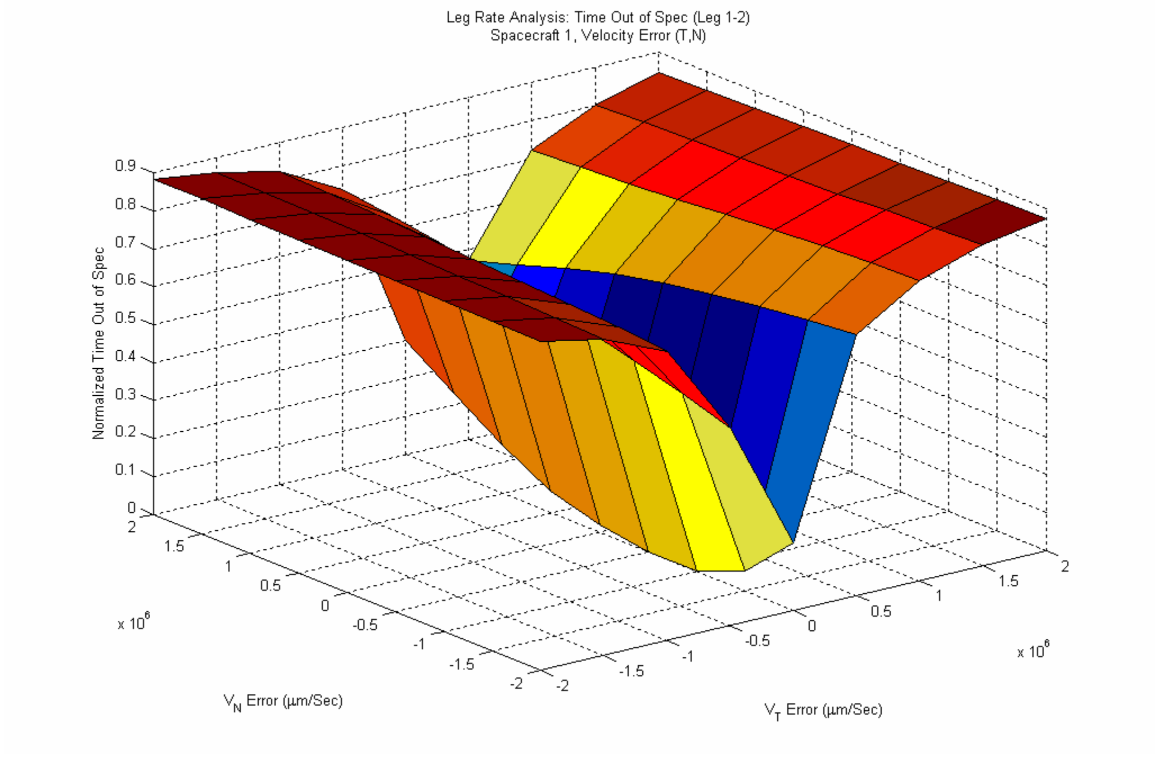


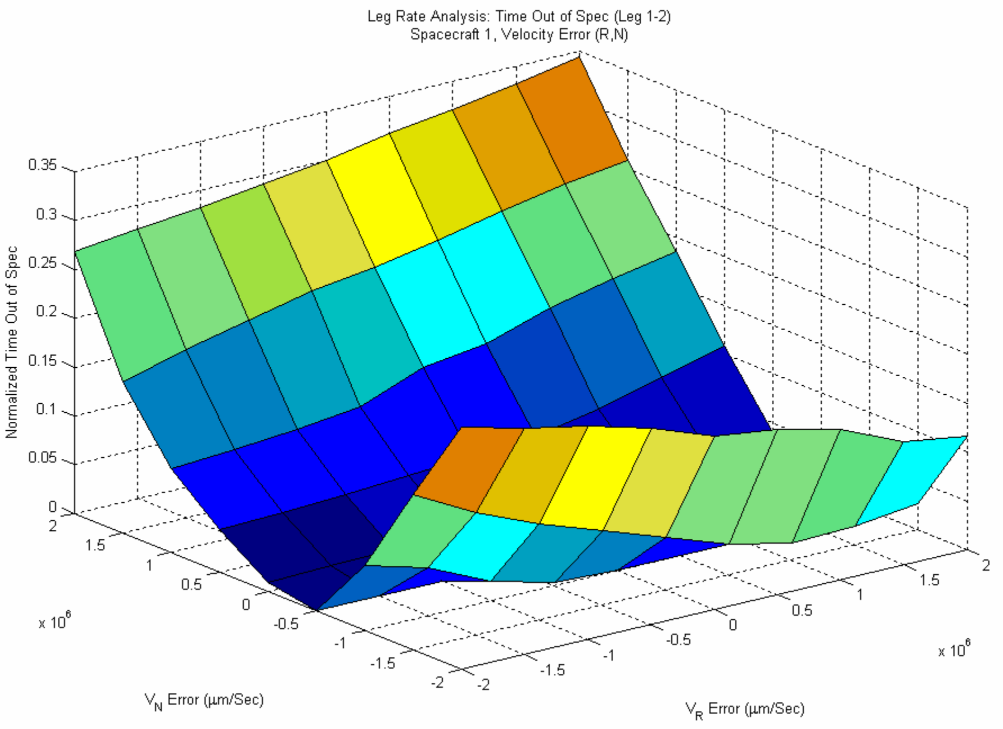

Leg Rate Analysis: Time Out of Spec (Leg 1-2)
Spacecraft 1, Position Error (R,N)
Normalized Time Out of Spec
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0.16
0.18
N Error (m)
x 10^5
-2
0
2
4
6
8
R Error (m)
x 10^5
-2
-1
0
1
2
3
4
5
6
7
8


Leg Rate Analysis: Time Out of Spec (Leg 1-2)
Spacecraft 1, Position Error (T,N)
Normalized Time Out of Spec
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
N Error (m)
x 10^5
-2
0
2
4
6
8
T Error (m)
x 10^5
-2
-1
0
1
2
3
4
5
6
7
8

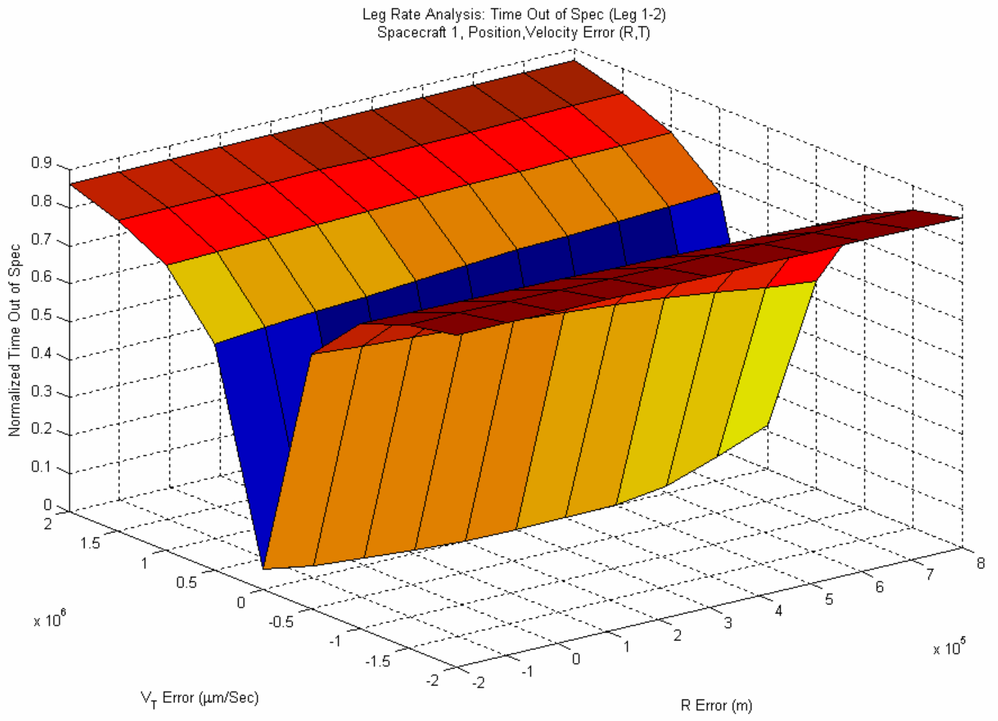
Velocity Errors Along Various Directions

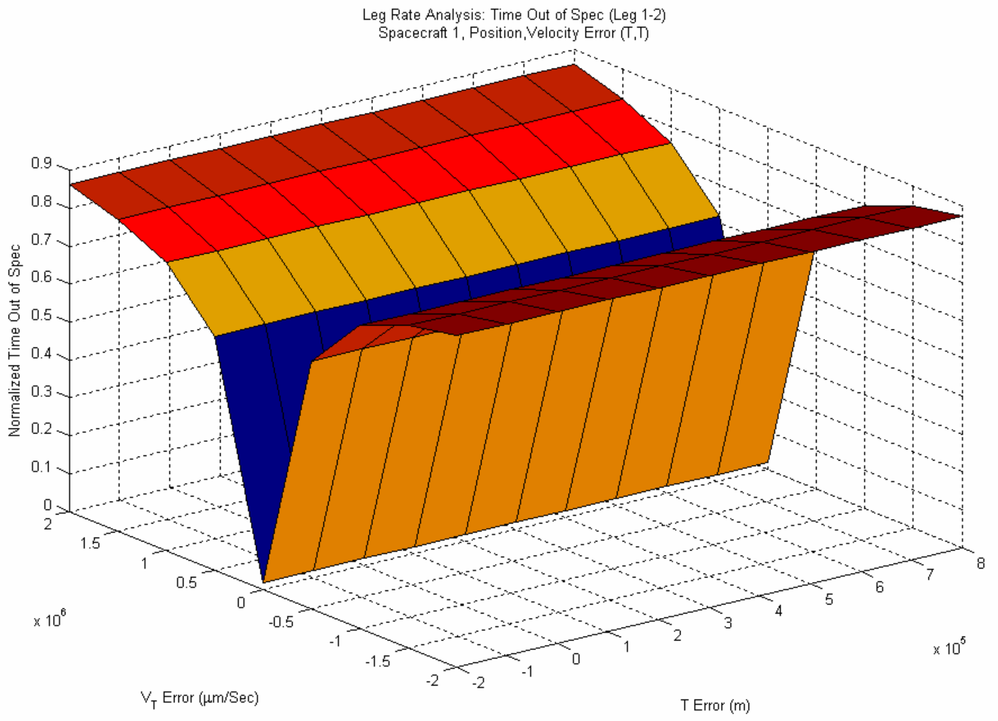


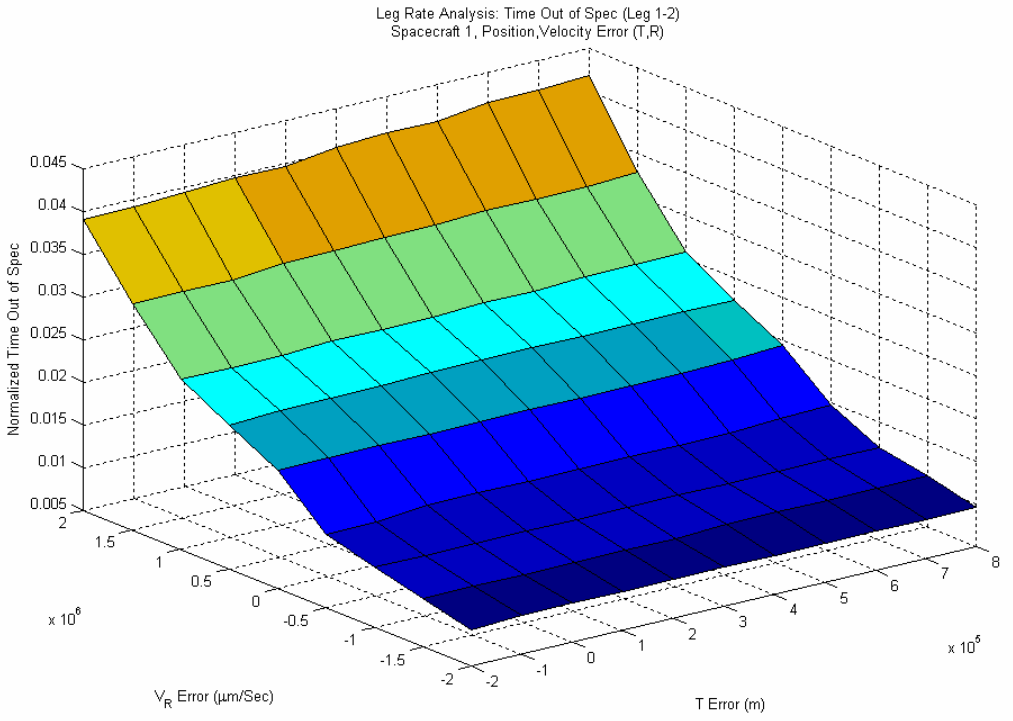


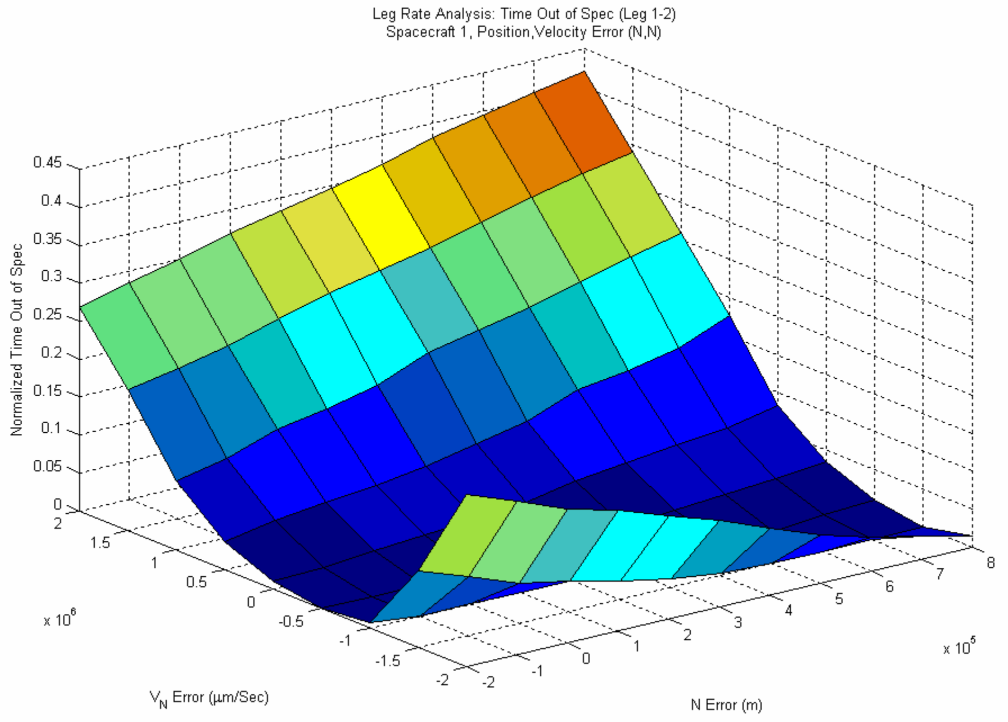

Leg Rate Analysis: Time Out of Spec (Leg 1-2)
Spacecraft 1, Velocity Error (R,N)
Normalized Time Out of Spec
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
x 10^6
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
V_N Error (μm/Sec)
V_R Error (μm/Sec)
x 10^6

Position-Velocity Errors Along Various Directions

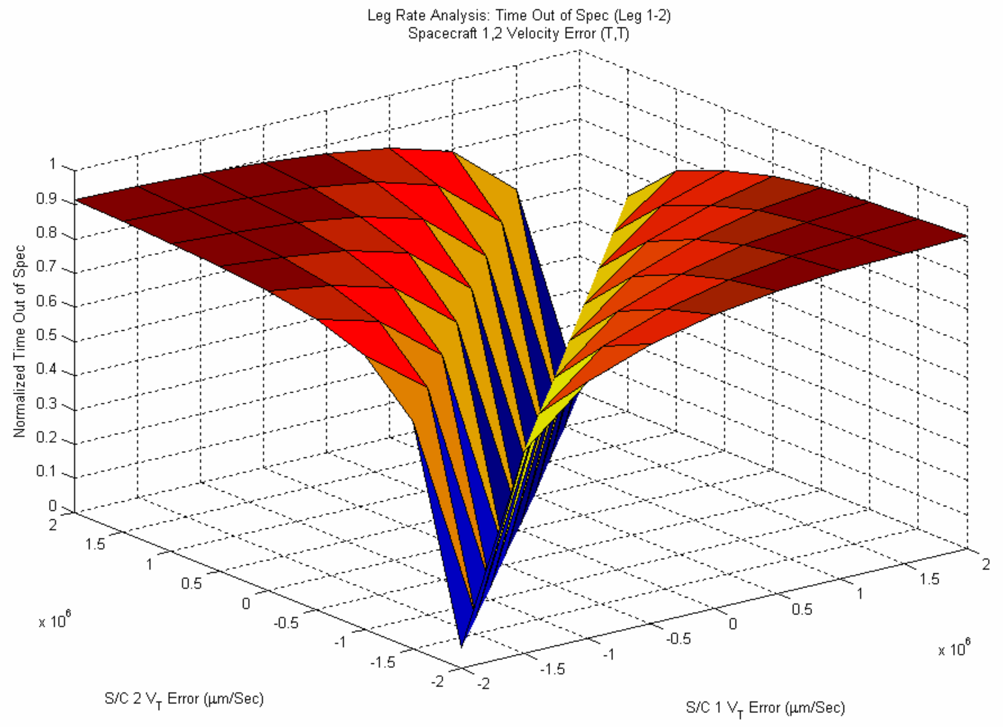


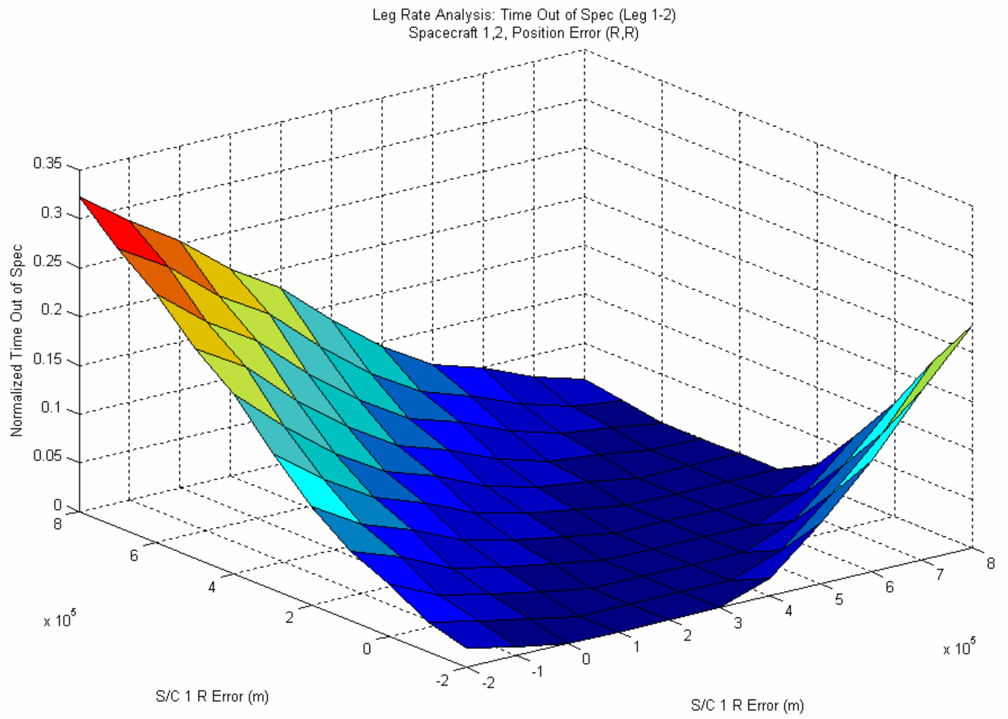



Leg Rate Analysis: Time Out of Spec (Leg 1-2)
Spacecraft 1, Position,Velocity Error (T,R)
Normalized Time Out of Spec
0.045
0.04
0.035
0.03
0.025
0.02
0.015
0.01
0.005
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
x 10^6
V_R Error (µm/Sec)
-2
-1
0
1
2
3
4
5
6
7
8
x 10^5
T Error (m)

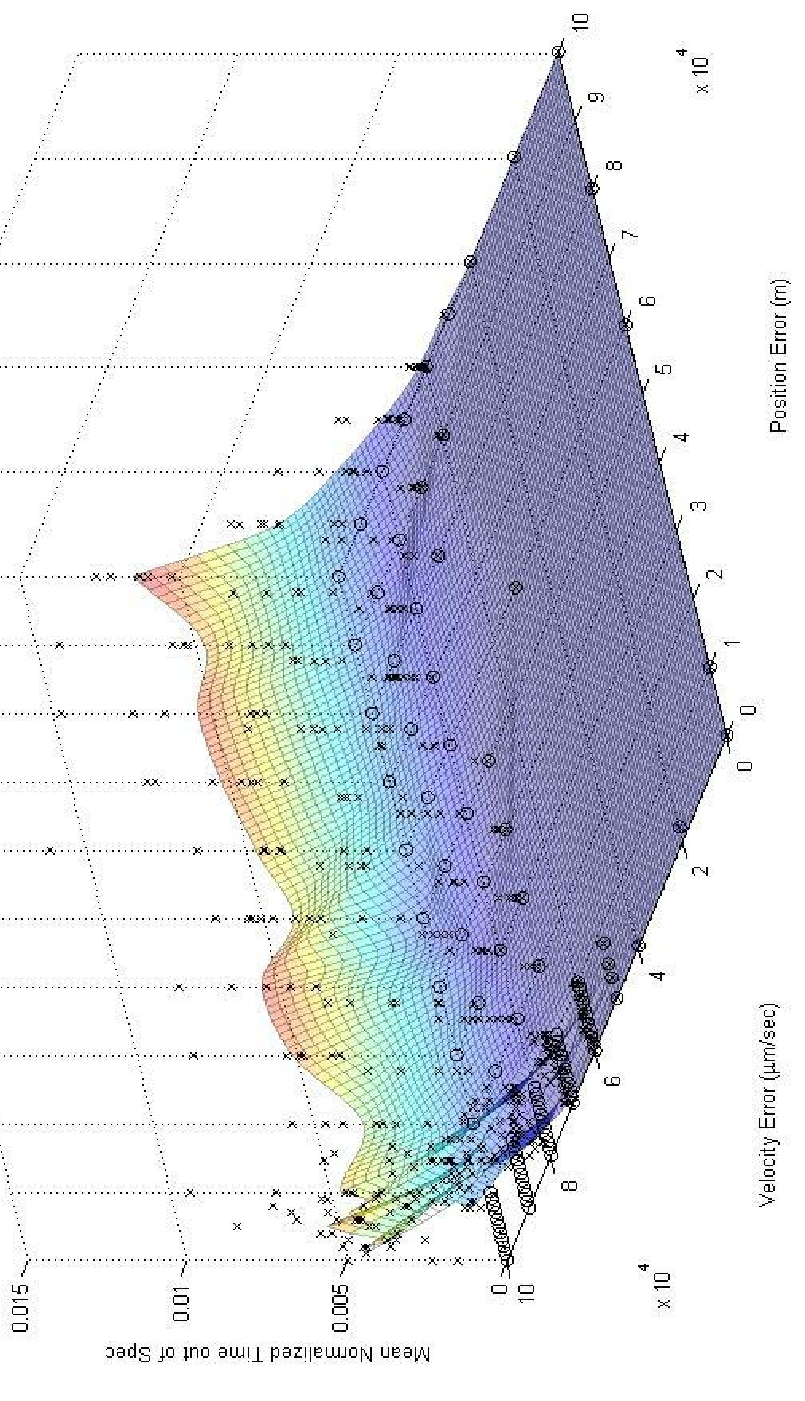

Leg Rate Analysis: Time Out of Spec (Leg 1-2)
Spacecraft 1, Position,Velocity Error (N,N)
Normalized Time Out of Spec
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
x 10^6
V_N Error (µm/Sec)
-2
-1
0
1
2
3
4
5
6
7
8
x 10^5
N Error (m)

Multi-Spacecraft Errors of Similar Type





Appendix B – Rainplot



Appendix C – Analysis Tools

Time Domain Analysis

This tool works by being given an insertion error scenario. It then checks the available orbit data to determine if any propagation data are needed. If so, it will write a file for the propagator containing the required propagations. When it finds the propagation data, the tool will then create an array containing each formation parameter as a function of time, which is then plotted on a series of graphs. The labels, titles, and limits for each graph are automatically generated to show the insertion error scenario, the limits for that particular formation parameter, and the units used.

Error Domain Analysis

Two Parameter Variation

A high and low limit for each axis is specified, along with the type and direction of error, and the spacecraft each affects. An array containing all combinations of possible initial error based on the given step is created. The tool then goes through the available propagation data to determine if any propagations are required. If so, it will write a file for the propagator containing the required propagations. When it finds all of the propagations, the tool will then run a time domain analysis for each error combination, then calculate the percent time out of specification for the leg length rate of change. It saves this information to a database. A surface plot is then formed with all labels and the title automatically generated.

Full Parameter Variation

Maximum position and velocity error magnitude are specified, as well as the number of formations per trial to be used. A set of random initial errors is created for each satellite for each formation and is then saved into a file for the propagator to read. Upon completion of all propagation, the data are analyzed in the time domain, and the time out of specification for the leg length rate of change parameter is determined. Several statistics from each run is calculated including the mean, maximum, minimum, and standard deviation. All data are then saved into a database.

To generate the plot, the database is read, x's and o's are plotted, and a surface is generated using cubic regression of the means of the type of plot specified (mean, maximum, minimum, or standard deviation).

Cross-section generation follows the same method as above, except the magnitude of position error is held constant, and no regression is performed on the plot. In addition, the plot is on a two-dimensional axis, not a three-dimensional one.

Appendix D – Analysis Code

Miscellaneous Scripts

Initialize.m

```
%this script initializes the project to enable all forms of analysis to
be
%run by setting up global variables and plot labels.

global errdata
phase1
global abscissa
global ordinate
SFAG
MCAG
global mcagarray
rainplot

labelarray = [{'R Error (m)'};...
    {'T Error (m)'};...
    {'N Error (m)'};...
    {'Vr Error (\mum/Sec)'};...
    {'Vt Error (\mum/Sec)'};...
    {'Vn Error (\mum/Sec)'}];

[LRAdbData,LRAdbData5yr] = LRAdbInit();
curDB = 0;
LRAdbTemp = [];
LRAtemp = [];
```

datafilename.m

```
%This function creates a structure with the 3 filenames for the
spacecraft
%propagation data

function [fn] = datafilename(errdata)

    dbdir = 'c:\\data\\';

    fn.sc1 = [dbdir
'sc1\\sc1_x_',int2str(errdata(1)),'_y_',int2str(errdata(4)),'_z_',int2s
tr(errdata(7)),...
'_vx_',int2str(errdata(10)),'_vy_',int2str(errdata(13)),'_vz_',int2str(
errdata(16))];

    fn.sc2 = [dbdir
'sc2\\sc2_x_',int2str(errdata(2)),'_y_',int2str(errdata(5)),'_z_',int2s
tr(errdata(8)),...
'_vx_',int2str(errdata(11)),'_vy_',int2str(errdata(14)),'_vz_',int2str(
errdata(17))];
```

```
    fn.sc3 = [dbdir
'sc3\\sc3_x_',int2str(errdata(3)),'_y_',int2str(errdata(6)),'_z_',int2s
tr(errdata(9)),...
'_vx_',int2str(errdata(12)),'_vy_',int2str(errdata(15)),'_vz_',int2str(
errdata(18))];
```

datacheck.m

```
%this function generates a flag value based on whether or not raw orbit
%data files exist

function [flag] = datacheck(fn)

    fn1id = fopen(fn.sc1);
    fn2id = fopen(fn.sc2);
    fn3id = fopen(fn.sc3);

    flag = 0;

    if fn1id == -1
        flag = 1;
    else
        fclose(fn1id);
    end
    if fn2id == -1
        flag = flag + 2;
    else
        fclose(fn2id);
    end
    if fn3id == -1
        flag = flag + 4;
    else
        fclose(fn3id);
    end
```

CurRunVar.m

```
%This function generates a string that is composed of
%sc1x_sc2x_sc3x_sc1y_etc...

function [CurRun] = CurRunVar(errdata)
    CurRun = [];
    for i = [1:17]
        CurRun = [CurRun,int2str(errdata(i)),'_'];
    end
    CurRun = [CurRun,int2str(errdata(18))];
```

md5.m

```
function y = md5( a1, a2, a3 );
%MD5 verifies or generates a signature using the md5 algorithm.
%   MD5( M ) or MD5( M, 0 ) returns a message digest (signature)
%   from the matrix M. Currently the classes double and char are
supported.
%
%   MD5( M, 1 )  generates the digest from a file. M must be a char
%   array with the filename/filepath.
```

```
%
%   You can also give a signature as the last argument. In this case
the
%   generated signature will be compared against the given. Returns 0
or 1.
%   Example: MD5( M, 1, '7dea362b3fac8e00956a4952a3d4f474' );
%
%   Md5 is actually not intended to work with large files (> 5 MB, see
notes),
%   but is really comfortable to process directly matlab matrices.


%   Notes:     o There are more hashing routines, that could be
implemented
%                eg. CRC, Adler, Haval, SHA, RMD...
%              o There's a problem with incremental file reading. As a
workaround
%                I had to load the whole file into the memory. I tested
with a 50 MB
%                file but though it worked well, I should fix this
problem if there's
%                a need to process large files.
%              o For questions/comments/requests: support@treetron.ch.
%
%   Credits:   I used a freeware library with different hash
algorithms. It's from
%              Alex? (Ritlabs) and was downloaded from Torrys. Thanks a
lot.
%              Built with Borland Delphi.
%
%   License:   You may use and distribute md5 free of charge for
commercial and
%              non-commercial use. Please don't modify this notice.
Before using this
%              routine you have to accept the disclaimer of warranty
below.
%
%   Warranty:  md5 is supplied as is. The author disclaims all
warranties,
%              expressed or implied, including, without limitation, the
warranties of
%              merchantability and of fitness for any purpose. The
author assumes no
%              liability for damages, direct or consequential, which
may result from the
%              use of md5.
%
%   Author:    Hans-Peter Suter
%   Revision:  0.7
%   Date:      25.7.2003
%
%   Copyright: Copyright (c) 2003, Treetron GmbH.
%              All rights reserved.

if nargin == 3
```

```
  b1 = a1; % matrix
  b2 = a2; % isFile
  b3 = a3; % signature
elseif nargin == 2
  if isa( a2, 'char' )
    b1 = a1;
    b2 = 0;
    b3 = a2;
  else
    b1 = a1;
    b2 = a2;
    b3 = [];
  end
elseif nargin == 1
  b1 = a1;
  b2 = 0;
  b3 = [];
else
  error( '3 arguments required' );
end;

% some checks
if ~isempty( b3 )
  if ~isa( b3, 'char' )
    error( 'signature must be a char array' );
  end
  if length( b3 ) ~= 32
    error( 'signature must have 32 chars' );
  end
end
if ~(isa( b1, 'char' ) | isa( b1, 'double' ))
  error( 'value/filename must be a double or char array' );
end
if ~(b2 == 0 | b2 == 1)
  error( 'isFile must be 0 or 1' );
end

% call dll
if isempty( b3 )
  y = md5DLL( b1, b2 );
else
  y = md5DLL( b1, b2, b3 );
end;
```

Time Domain Analysis Scripts

maganalysis.m

```
%This script determines if the orbit data exists,
%then runs the actual analysis script, FormAnal()

fn = datafilename(errdata);
flag = datacheck(fn);

if ~flag
```

```
    sc1 = dlmread(fn.sc1,',');
    sc2 = dlmread(fn.sc2,',');
    sc3 = dlmread(fn.sc3,',');

    magsc = FormAnal(sc1,sc2,sc3);
    fna = 'SC1_';
    fna = [fna,num2str(errdata(1)) '_'];
    fna = [fna,num2str(errdata(4)) '_'];
    fna = [fna,num2str(errdata(7)) '_'];
    fna = [fna,num2str(errdata(10)) '_'];
    fna = [fna,num2str(errdata(13)) '_'];
    fna = [fna,num2str(errdata(16)) '; SC2_'];
    fna = [fna,num2str(errdata(2)) '_'];
    fna = [fna,num2str(errdata(5)) '_'];
    fna = [fna,num2str(errdata(8)) '_'];
    fna = [fna,num2str(errdata(11)) '_'];
    fna = [fna,num2str(errdata(14)) '_'];
    fna = [fna,num2str(errdata(17)) '; SC3_'];
    fna = [fna,num2str(errdata(3)) '_'];
    fna = [fna,num2str(errdata(6)) '_'];
    fna = [fna,num2str(errdata(9)) '_'];
    fna = [fna,num2str(errdata(12)) '_'];
    fna = [fna,num2str(errdata(15)) '_'];
    fna = [fna,num2str(errdata(18))];

    %legrateanalyzer
    %LRAdbSave
    %analysisplotter
    %lldpplotter

else

    dlmwrite('c:\\pyshell\\scripts\\LISA\\runs.tsp',errdata,'-append')
    if (flag == 1) || (flag == 3) || (flag == 7) || (flag == 5)
        disp([fn.sc1,' does not exist'])
    end
    if (flag == 2) || (flag == 3) || (flag == 6) || (flag == 7)
        disp([fn.sc2,' does not exist'])
    end
    if (flag == 4) || (flag == 5) || (flag == 6) || (flag == 7)
        disp([fn.sc3,' does not exist'])
    end
end
```

FormAnal.m

```
%this function analyzes the raw orbital data files to generate the
%different formation parameters.

function [magsc] = FormAnal(sc1,sc2,sc3)

    earthbuff = dlmread('e:\\data\\earthvec',',');
    earthvec =
[earthbuff(:,1),earthbuff(:,2),earthbuff(:,3),earthbuff(:,4)];
```

```
    sc12=sc2(:,2:4)-sc1(:,2:4);%km
    sc13=sc3(:,2:4)-sc1(:,2:4);
    sc23=sc3(:,2:4)-sc2(:,2:4);
    sc12v=1000.*(sc2(:,5:7)-sc1(:,5:7));%m/s
    sc13v=1000.*(sc3(:,5:7)-sc1(:,5:7));
    sc23v=1000.*(sc3(:,5:7)-sc2(:,5:7));

    magsc12 = sqrt(sum(sc12'.^2)');%km
    magsc13 = sqrt(sum(sc13'.^2)');
    magsc23 = sqrt(sum(sc23'.^2)');
    magsc12v = sqrt(sum(sc12v'.^2)');%m/s
    magsc13v = sqrt(sum(sc13v'.^2)');
    magsc23v = sqrt(sum(sc23v'.^2)');
    dmagsc12 = (dot(1000.*sc12',sc12v')'./(1000.*magsc12));%m/s
    dmagsc13 = (dot(1000.*sc13',sc13v')'./(1000.*magsc13));
    dmagsc23 = (dot(1000.*sc23',sc23v')'./(1000.*magsc23));

    angle1 = acosd(dot(sc12',sc13')'./(magsc12.*magsc13));%deg
    angle2 = acosd(-dot(sc12',sc23')'./(magsc12.*magsc23));
    angle3 = acosd(dot(sc23',sc13')'./(magsc23.*magsc13));

    angle1v = (dot(sc12v',1000.*sc13')'+dot(1000.*sc12',sc13v')'-
(dmagsc12.*1000.*magsc13+1000.*magsc12.*dmagsc13).*cosd(angle1));
    angle1v = angle1v./(-1000.*magsc12.*1000.*magsc13.*sind(angle1));

    angle2v = (dot(-sc12v',1000.*sc23')'+dot(1000.*-sc12',sc23v')'-
(dmagsc12.*1000.*magsc23+1000.*magsc12.*dmagsc23).*cosd(angle2));
    angle2v = angle2v./(-1000.*magsc12.*1000.*magsc23.*sind(angle2));
%angle2v = 0.*angle2v;
    angle3v = (dot(sc23v',sc13')'+dot(sc23',sc13v')'-
(dmagsc23.*magsc13+magsc23.*dmagsc13).*cosd(angle3));
    angle3v = angle3v./(-magsc23.*magsc13.*sind(angle3));
    angle3v = 0.*angle3v;

    scc = (sc1(:,2:4)+sc2(:,2:4)+sc3(:,2:4))./3;
    earthpos = earthvec(:,2:4);
    magscc = sqrt(sum(scc'.^2)');
    magearthpos = sqrt(sum(earthpos'.^2)');
    earthangle = acosd(dot(scc',earthpos')'./(magscc.*magearthpos));


    magsc=[magsc12,magsc13,magsc23,...
        dmagsc12,dmagsc13,dmagsc23,...
        angle1,angle2,angle3,...
        angle1v,angle2v,angle3v,...
        earthangle];
```

analysisplotter.m

```
%This script plots the various formation parameters WRT time

newfig = figure;

plot(magsc(:,1:3))
title(['Leg Length vs Time Plot (',fna,')'],'interpreter','none')
xlabel('Time from epoch (days)')
```

```
ylabel('Leg Length (Km)')
legend('1-2','1-3','2-3')
line(xlim,[5.05e6 5.05e6],'Color','k')
line(xlim,[4.95e6 4.95e6],'Color','k')
saveas(newfig,['c:\\tsptemp\\LLP',fna(9:length(fna))],'fig')

newfig = figure;

plot(magsc(:,4:6))
title(['Leg Length Rate of Change vs Time Plot
(',fna,')'],'interpreter','none')
xlabel('Time from epoch (days)')
ylabel('Leg Length Rate of Change (m/sec)')
legend('1-2','1-3','2-3')
line(xlim,[15 15],'Color','k')
line(xlim,[-15 -15],'Color','k')
saveas(newfig,['c:\\tsptemp\\LLROCP',fna(9:length(fna))],'fig')

newfig = figure;

plot(magsc(:,7:9))
title(['Leg Angle Plot (',fna,')'],'interpreter','none')
xlabel('Time from epoch (days)')
ylabel('Leg Angle (deg)')
legend('Vertex 1','Vertex 2','Vertex 3')
line(xlim,[61.5 61.5],'Color','k')
line(xlim,[58.5 58.5],'Color','k')
saveas(newfig,['c:\\tsptemp\\LAP',fna(9:length(fna))],'fig')



newfig = figure;

plot(magsc(:,13))
title(['Earth Angle Plot (',fna,')'],'interpreter','none')
xlabel('Time from epoch (days)')
ylabel('Earth Angle (deg)')
line(xlim,[25 25],'Color','k')
line(xlim,[20 20],'Color','k')
saveas(newfig,['c:\\tsptemp\\EAP',fna(9:length(fna))],'fig')
```

baselineplots.m

```
%this script plots the baseline formation parameters WRT time

format long
format compact

errdata = zeros(1,18);

fn = datafilename(errdata);

sc1 = dlmread(fn.sc1,',');
sc2 = dlmread(fn.sc2,',');
sc3 = dlmread(fn.sc3,',');
```

```
earthbuff = dlmread('e:\\data\\earthvec',',');
earth = [earthbuff(:,1),earthbuff(:,2),earthbuff(:,3),earthbuff(:,4)];

sc12=sc2(:,2:4)-sc1(:,2:4);%km
sc13=sc3(:,2:4)-sc1(:,2:4);
sc23=sc3(:,2:4)-sc2(:,2:4);
sc12v=1000.*(sc2(:,5:7)-sc1(:,5:7));%m/s
sc13v=1000.*(sc3(:,5:7)-sc1(:,5:7));
sc23v=1000.*(sc3(:,5:7)-sc2(:,5:7));

magsc12 = sqrt(sum(sc12'.^2)');%km
magsc13 = sqrt(sum(sc13'.^2)');
magsc23 = sqrt(sum(sc23'.^2)');
magsc12v = sqrt(sum(sc12v'.^2)');%m/s
magsc13v = sqrt(sum(sc13v'.^2)');
magsc23v = sqrt(sum(sc23v'.^2)');
dmagsc12 = (dot(1000.*sc12',sc12v')'./(1000.*magsc12));%m/s
dmagsc13 = (dot(1000.*sc13',sc13v')'./(1000.*magsc13));
dmagsc23 = (dot(1000.*sc23',sc23v')'./(1000.*magsc23));

angle1 = acosd(dot(sc12',sc13')'./(magsc12.*magsc13));%deg
angle2 = acosd(-dot(sc12',sc23')'./(magsc12.*magsc23));
angle3 = acosd(dot(sc23',sc13')'./(magsc23.*magsc13));

angle1v = (dot(sc12v',1000.*sc13')'+dot(1000.*sc12',sc13v')'-
(dmagsc12.*1000.*magsc13+1000.*magsc12.*dmagsc13).*cosd(angle1));
angle1v = angle1v./(-1000.*magsc12.*1000.*magsc13.*sind(angle1));

angle2v = (dot(-sc12v',1000.*sc23')'+dot(1000.*-sc12',sc23v')'-
(dmagsc12.*1000.*magsc23+1000.*magsc12.*dmagsc23).*cosd(angle2));
angle2v = angle2v./(-1000.*magsc12.*1000.*magsc23.*sind(angle2));
%angle2v = 0.*angle2v;
angle3v = (dot(sc23v',sc13')'+dot(sc23',sc13v')'-
(dmagsc23.*magsc13+magsc23.*dmagsc13).*cosd(angle3));
angle3v = angle3v./(-magsc23.*magsc13.*sind(angle3));
angle3v = 0.*angle3v;

%angle1v = acosd(dot(sc12v',sc13v')'./(magsc12v.*magsc13v));
%angle2v = acosd(-dot(sc12v',sc23v')'./(magsc12v.*magsc23v));
%angle3v = acosd(dot(sc23v',sc13v')'./(magsc23v.*magsc13v));

scc = (sc1(:,2:4)+sc2(:,2:4)+sc3(:,2:4))./3;
earthpos = earth(:,2:4);
magscc = sqrt(sum(scc'.^2)');
magearthpos = sqrt(sum(earthpos'.^2)');
earthangle = acosd(dot(scc',earthpos')'./(magscc.*magearthpos));


magsc=[magsc12,magsc13,magsc23,...
    dmagsc12,dmagsc13,dmagsc23,...
    angle1,angle2,angle3,...
    angle1v,angle2v,angle3v,...
    earthangle];

newfig = figure;
```

```
plot(magsc(:,1:3))
title(['Leg Length vs Time Plot (baseline)'],'interpreter','none')
xlabel('Time from epoch (days)')
ylabel('Leg Length (Km)')
legend('1-2','1-3','2-3')
line(xlim,[5.05e6 5.05e6],'Color','k')
line(xlim,[4.95e6 4.95e6],'Color','k')

newfig = figure;

plot(magsc(:,4:6))
title(['Leg Length Rate of Change vs Time Plot
(baseline)'],'interpreter','none')
xlabel('Time from epoch (days)')
ylabel('Leg Length Rate of Change (m/sec)')
legend('1-2','1-3','2-3')
line(xlim,[15 15],'Color','k')
line(xlim,[-15 -15],'Color','k')

newfig = figure;

plot(magsc(:,7:9))
title(['Leg Angle Plot (baseline)'],'interpreter','none')
xlabel('Time from epoch (days)')
ylabel('Leg Angle (deg)')
legend('Vertex 1','Vertex 2','Vertex 3')
line(xlim,[61.5 61.5],'Color','k')
line(xlim,[58.5 58.5],'Color','k')

newfig = figure;

plot(magsc(:,10:12))
title(['Leg Angle Rate of Change Plot
(baseline)'],'interpreter','none')
xlabel('Time from epoch (days)')
ylabel('Leg Angle Rate of Change (deg/sec)')
legend('Vertex 1','Vertex 2','Vertex 3')

newfig = figure;

plot(magsc(:,13))
title(['Earth Angle Plot (baseline)'],'interpreter','none')
xlabel('Time from epoch (days)')
ylabel('Earth Angle (deg)')
line(xlim,[25 25],'Color','k')
line(xlim,[20 20],'Color','k')
```

Error Domain Analysis – Two Parameter Variation Scripts

SFDG.m

```
%generates data for the two-parameter variation plots
x = abscissa(3):abscissa(4):abscissa(5);
y = ordinate(3):ordinate(4):ordinate(5);
```

```
run = 1;
deltas = [];
for a = x
    for b = y
        deltas(run).errdata = zeros(1,18);
        deltas(run).errdata(3*abscissa(2)-(3-abscissa(1))) = a;
        deltas(run).errdata(3*ordinate(2)-(3-ordinate(1))) = b;
        run = run+1;
    end
end

proplist = [];

for a = deltas
    errdata = a.errdata;
    fn = datafilename(errdata);
    flag = datacheck(fn);
    if (flag)
        proplist = [proplist;errdata];
    elseif isempty(LRAdbLoad(errdata,LRAdbData))
        sc1 = dlmread(fn.sc1,',');
        sc2 = dlmread(fn.sc2,',');
        sc3 = dlmread(fn.sc3,',');
        magsc = FormAnal(sc1,sc2,sc3);
        try
            legrateanalyzer
        catch
            disp('Error in Analysis, exiting script without saving...')
            return
        end
        LRAdbTempSave
    end
end

if ~isempty(proplist)
    dlmwrite('c:\\pyshell\\scripts\\LISA\\runs.tsp',proplist,'-
append','precision','%20.0f')
    disp( ' ' )
    disp( ['There are ' num2str(length(proplist)) ' Propagations
Required.'] )
    disp( ' ' )
else
    disp(' ')
    disp('No Propagations Required, saving.')
    disp(' ')
    if ~isempty(LRAtemp)
        LRAdbSave
    end
end
```

LRAdbLoad.m

```
%Retrieves data from the LRA database

function CurRunStruct = LRAdbLoad(errdata,LRAdbData)
```

```
    currentrun = CurRunVar(errdata);
    currentrun = ['LRA_' md5(currentrun)];
    disp([CurRunVar(errdata) ' -> ' currentrun])

        if isfield(LRAdbData,currentrun)
            CurRunStruct = LRAdbData.(currentrun);
        else
            CurRunStruct = [];
        end
```

legrateanalyzer.m

```
%analyzes time out of spec for leg rate parameter over 10 years

highlim = 15;
lowlim = -15;

%magsc=[magsc12,magsc13,magsc23,...
%        dmagsc12,dmagsc13,dmagsc23,...
%        angle1,angle2,angle3,...
%        angle1v,angle2v,angle3v,...
%        earthangle];

binarray12 = (magsc(:,4) > highlim) | (magsc(:,4) < lowlim);
binarray13 = (magsc(:,5) > highlim) | (magsc(:,5) < lowlim);
binarray23 = (magsc(:,6) > highlim) | (magsc(:,6) < lowlim);

binarrayloc12 = find(binarray12);
binarrayloc13 = find(binarray13);
binarrayloc23 = find(binarray23);

if ~isempty(binarrayloc12)
    dbinarrayloc12 = diff(binarrayloc12) - 1;
    pulsestart12 = binarrayloc12(1);
    pulseend12 = binarrayloc12(length(binarrayloc12));
    spikes12 = find(dbinarrayloc12);
    numspikes12 = length(spikes12);
    if numspikes12 >= 1
        pulsestart12 =
[pulsestart12;binarrayloc12(find(dbinarrayloc12)+1)];
        pulseend12 = [binarrayloc12(find(dbinarrayloc12));pulseend12];
    end
else
    pulsestart12 = [];
    pulseend12 = [];
    spikes12 = [];
end

if ~isempty(binarrayloc13)
    dbinarrayloc13 = diff(binarrayloc13) - 1;
    pulsestart13 = binarrayloc13(1);
    pulseend13 = binarrayloc13(length(binarrayloc13));
    spikes13 = find(dbinarrayloc13);
    numspikes13 = length(find(diff(binarrayloc13)-1));
    if numspikes13 >= 1
```

```
        pulsestart13 =
[pulsestart13;binarrayloc13(find(dbinarrayloc13)+1)];
        pulseend13 = [binarrayloc13(find(dbinarrayloc13));pulseend13];
    end
else
    pulsestart13 = [];
    pulseend13 = [];
    spikes13 = [];
end

if ~isempty(binarrayloc23)
    dbinarrayloc23 = diff(binarrayloc23) - 1;
    pulsestart23 = binarrayloc23(1);
    pulseend23 = binarrayloc23(length(binarrayloc23));
    spikes23 = find(dbinarrayloc23-1);
    numspikes23 = length(dbinarrayloc23);
    if numspikes23 >= 1
        pulsestart23 =
[pulsestart23;binarrayloc23(find(dbinarrayloc23)+1)];
        pulseend23 = [binarrayloc23(find(dbinarrayloc23));pulseend23];
    end
else
    pulsestart23 = [];
    pulseend23 = [];
    spikes23 = [];
end
```

LRAdbTempSave.m

```
%LRAdbTempSave - saves the legrateanalyzer output to a temporary
structure

dbdir = 'c:\\data\\LRAdb\\';

currentrun = CurRunVar(errdata);
currentrun = ['LRA_' md5(currentrun)];
disp([currentrun]);

LRAtemp.(currentrun).pulsestart12 = pulsestart12;
LRAtemp.(currentrun).pulseend12 = pulseend12;
LRAtemp.(currentrun).pulsestart13 = pulsestart13;
LRAtemp.(currentrun).pulseend13 = pulseend13;
LRAtemp.(currentrun).pulsestart23 = pulsestart23;
LRAtemp.(currentrun).pulseend23 = pulseend23;
LRAtemp.(currentrun).spikes12 = spikes12;
LRAtemp.(currentrun).spikes13 = spikes13;
LRAtemp.(currentrun).spikes23 = spikes23;
```

LRAdbSave.m

```
%LRAdbSave - saves the temporary structure to the LRA database

dbdir = 'c:\\data\\LRAdb\\';

currentrun = CurRunVar(errdata);
currentrun = ['LRA_' md5(currentrun)];
```

```
buff = fieldnames(LRAtemp);
for a =1:length(buff)
    LRAdbData.(char(buff(a))) = LRAtemp.(char(buff(a)));
end

save([dbdir 'LRAdb.mat'],'LRAdbData');
```

LRATPctGraph.m

```
%LRATPctGraph - Plots the two-parameter variation data

b = 1;
c = 1;
run = 1;
z = zeros(length(y),length(x));
for a = deltas
    data = LRAdbLoad(a.errdata,LRAdbData);
    z(c,b) = sum(data.pulseend12 - data.pulsestart12)/3654;
    c = c+1;
    if c == (length(y)+1)
        b = b+1;
        c = 1;
    end
end
surf(x,y,z);
xlabel(labelarray(abscissa(2)))
ylabel(labelarray(ordinate(2)))
zlabel('Normalized Time Out of Spec')
```

Error Domain Analysis – Full Parameter Variation Scripts

MCDG1.m

```
% Monte Carlo Data Generator part 1

rand('state', sum(100*clock));
errdataruns = rand(mcagarray(1),18) - 0.5;
errdataruns(:,1:9) = round(mcagarray(2)*errdataruns(:,1:9));
errdataruns(:,10:18) = round(mcagarray(3)*errdataruns(:,10:18));

dlmwrite('c:\\pyshell\\scripts\\LISA\\runs.tsp',errdataruns,'-append')
```

MCDG25yr.m

```
MCDG2LRA5yr
clear errdataruns
```

MCDG2LRA5yr.m

```
%Monte Carlo Data Generator part 2

PooSraw = [];

for i = 1:mcagarray(1)
    errdata = errdataruns(i,:);
```

```
        fn = datafilename(errdata);
        sc1 = dlmread(fn.sc1,',');
        sc2 = dlmread(fn.sc2,',');
        sc3 = dlmread(fn.sc3,',');
        magsc = FormAnal(sc1,sc2,sc3);
        legrateanalyzer5yr
        currentrun = ['LRA_' md5(CurRunVar(errdata))];
        disp(currentrun)

    data.pulsestart12 = pulsestart12;
    data.pulseend12 = pulseend12;
    data.pulsestart13 = pulsestart13;
    data.pulseend13 = pulseend13;
    data.pulsestart23 = pulsestart23;
    data.pulseend23 = pulseend23;
    data.spikes12 = spikes12;
    data.spikes13 = spikes13;
    data.spikes23 = spikes23;

    binarray12 = zeros(1,1827);
    binarray13 = zeros(1,1827);
    binarray23 = zeros(1,1827);
    for i = 1:length(data.pulsestart12)
        binarray12(data.pulsestart12(i):data.pulseend12(i)) = 1;
    end
    for i = 1:length(data.pulsestart13)
        binarray13(data.pulsestart13(i):data.pulseend13(i)) = 1;
    end
    for i = 1:length(data.pulsestart23)
        binarray23(data.pulsestart23(i):data.pulseend23(i)) = 1;
    end

    PooSraw = [PooSraw,sum((binarray12 | binarray13) | binarray23) /
1827];

end

PooSmean = mean(PooSraw);
PooSstd = std(PooSraw);
PooSmax = max(PooSraw);
PooSmin = min(PooSraw);

PooSdata = LRAPooSLoad5yr(mcagarray);

if isempty(PooSdata)
    PooSdata.mean = PooSmean;
    PooSdata.std = PooSstd;
    PooSdata.max = PooSmax;
    PooSdata.min = PooSmin;
else
    PooSdata.mean = [PooSdata.mean,PooSmean];
    PooSdata.std = [PooSdata.std,PooSstd];
    if PooSmax > PooSdata.max
        PooSdata.max = PooSmax;
    end
    if PooSmin < PooSdata.min
        Poosdata.min = PooSmin;
    end
end
LRAPooSSave5yr
```

legrateanalyzer5yr.m

```
%analyzes time out of spec for leg rate parameter over 5 years
highlim = 15;
lowlim = -15;

%magsc=[magsc12,magsc13,magsc23,...
%        dmagsc12,dmagsc13,dmagsc23,...
%        angle1,angle2,angle3,...
%        angle1v,angle2v,angle3v,...
%        earthangle];

binarray12 = (magsc(1:1827,4) > highlim) | (magsc(1:1827,4) < lowlim);
binarray13 = (magsc(1:1827,5) > highlim) | (magsc(1:1827,5) < lowlim);
binarray23 = (magsc(1:1827,6) > highlim) | (magsc(1:1827,6) < lowlim);

binarrayloc12 = find(binarray12);
binarrayloc13 = find(binarray13);
binarrayloc23 = find(binarray23);

if ~isempty(binarrayloc12)
    dbinarrayloc12 = diff(binarrayloc12) - 1;
    pulsestart12 = binarrayloc12(1);
    pulseend12 = binarrayloc12(length(binarrayloc12));
    spikes12 = find(dbinarrayloc12);
    numspikes12 = length(spikes12);
    if numspikes12 >= 1
        pulsestart12 =
[pulsestart12;binarrayloc12(find(dbinarrayloc12)+1)];
        pulseend12 = [binarrayloc12(find(dbinarrayloc12));pulseend12];
    end
else
    pulsestart12 = [];
    pulseend12 = [];
    spikes12 = [];
end

if ~isempty(binarrayloc13)
    dbinarrayloc13 = diff(binarrayloc13) - 1;
    pulsestart13 = binarrayloc13(1);
    pulseend13 = binarrayloc13(length(binarrayloc13));
    spikes13 = find(dbinarrayloc13);
    numspikes13 = length(find(diff(binarrayloc13)-1));
    if numspikes13 >= 1
        pulsestart13 =
[pulsestart13;binarrayloc13(find(dbinarrayloc13)+1)];
        pulseend13 = [binarrayloc13(find(dbinarrayloc13));pulseend13];
    end
else
```

```
    pulsestart13 = [];
    pulseend13 = [];
    spikes13 = [];
end

if ~isempty(binarrayloc23)
    dbinarrayloc23 = diff(binarrayloc23) - 1;
    pulsestart23 = binarrayloc23(1);
    pulseend23 = binarrayloc23(length(binarrayloc23));
    spikes23 = find(dbinarrayloc23-1);
    numspikes23 = length(dbinarrayloc23);
    if numspikes23 >= 1
        pulsestart23 =
[pulsestart23;binarrayloc23(find(dbinarrayloc23)+1)];
        pulseend23 = [binarrayloc23(find(dbinarrayloc23));pulseend23];
    end
else
    pulsestart23 = [];
    pulseend23 = [];
    spikes23 = [];
end
```

LRAPooSLoad5yr.m

```
%loads specific rainplot data from the LRAPooS (Percent out of Spec)
%database

function [data] = LRAPooSLoad(mcagarray)

dbdir = 'c:\\data\\LRAdb\\';

load([dbdir 'LRAPooS5yr.mat']);
nstr = ['n_' num2str(mcagarray(1))];
pstr = ['p_' num2str(mcagarray(2))];
vstr = ['v_' num2str(mcagarray(3))];

if isfield(LRAPooSData, nstr) & isfield(LRAPooSData.(nstr), pstr) &
isfield(LRAPooSData.(nstr).(pstr), vstr)
    data = LRAPooSData.(nstr).(pstr).(vstr);
else
    data = [];
end
```

LRAPooSSave5yr.m

```
%LRAPooSSave - saves data into the rainplot LRAPooS database

dbdir = 'c:\\data\\LRAdb\\';

load([dbdir 'LRAPooS5yr.mat']);
nstr = ['n_' num2str(mcagarray(1))];
pstr = ['p_' num2str(mcagarray(2))];
vstr = ['v_' num2str(mcagarray(3))];

LRAPooSData.(nstr).(pstr).(vstr) = PooSdata;
save([dbdir 'LRAPooS5yr.mat'],'LRAPooSData');
```

Rainplotter.m

```
%creates the actual rainplot
function [ state ] =
Rainplotter(dbdir,dbfile,db,ncurr,rplottitle,plim,vlim,type)

load([dbdir dbfile]);

eval(['narray = fieldnames(',db,');']);
eval(['parray = fieldnames(',db,'.(ncurr));']);

p = [];
v = [];
z = [];
for i = 1:length(parray)
    pcurr = char(parray(i));
    pcurrnum = str2num(pcurr(3:length(pcurr)));
    eval(['varray = fieldnames(',db,'.(ncurr).(pcurr));']);
    for j = 1:length(varray)
        vcurr = char(varray(j));
        vcurrnum = str2num(vcurr(3:length(vcurr)));
        eval(['z = [z,',db,'.(ncurr).(pcurr).(vcurr).',type,'];']);
        eval(['n = length(',db,'.(ncurr).(pcurr).(vcurr).',type,');']);
        p = [p,pcurrnum.*ones(1,n)];
        v = [v,vcurrnum.*ones(1,n)];
    end
end

figure

plot3(p,v,z,'xk',p,v,zeros(length(p),length(v)),'ok'),hold

if plim
    xlim([0 plim])
else
    plim = xlim;
    plim = plim(2);
end

if vlim
    ylim([0 vlim])
else
    vlim = xlim;
    vlim = vlim(2);
end

tx = 0:plim/100:plim;
ty = 0:vlim/100:vlim;
[XI,YI] = meshgrid(tx,ty);
ZI = griddata(p,v,z,XI,YI,'cubic');
surf(XI,YI,ZI,'FaceAlpha',0.33,'EdgeAlpha',0.33),hold off
zmin = zlim;
if zmin(1) < 0
    zlim([0 zmin(2)]);
end
grid on
```

```
xlabel('Position Error (m)')
ylabel('Velocity Error ({\mu}m/sec)')
zlabel('Mean Normalized Time out of Spec')
title(rplottitle)
```

Error Domain Analysis – Full Variation Rainplot Cross-Section Scripts

initialize.m

```
%this script initializes the project to enable all forms of analysis to
be
%run for rplotcross-section

global errdata
MCAG
global mcagarray
```

MCDG1b.m

```
% Monte Carlo Data Generator part 1 - rplotcross-section

rand('state', sum(100*clock));

bigrun = [];
bigmcagarray = [];
vvar = [40000:3000:70000];

for i = 1:length(vvar)
    for j = 1:6
        bigmcagarray = [bigmcagarray;mcagarray(1),0,vvar(i)];
        errdataruns = rand(mcagarray(1),18) - 0.5;
        errdataruns(:,1:9) = round(0*errdataruns(:,1:9));
        errdataruns(:,10:18) = round(vvar(i)*errdataruns(:,10:18));
        bigrun = [bigrun;errdataruns];
    end
end

dlmwrite('c:\\pyshell\\scripts\\LISA\\runs.tsp',bigrun,'-append')
dlmwrite('c:\\data\\temp\\batchruns.tsp',bigrun,'-append')
dlmwrite('c:\\data\\temp\\batchv.tsp',bigmcagarray,'-append')
```

MCDG25yrb.m

```
%modified from rplotcross-section

bigmcagarray = load('c:\\data\\temp\\batchv.tsp');
bigrun = load('c:\\data\\temp\\batchruns.tsp');
mcagarray(3) = 0;
[m,n] = size(bigmcagarray);
for amyers = 1:m
    mcagarray = bigmcagarray(amyers,:);
    errdataruns = bigrun(1+50*(amyers-1):50+50*(amyers-1),:);
    MCDG2LRA5yrb
    disp('------------')
    disp(['Run ' num2str(amyers) ' of ' num2str(m) ' complete.'])
    disp('------------')
end
clear errdataruns
clear varray
clear bigrun
delete('c:\\data\\temp\\batchv.tsp');
delete('c:\\data\\temp\\batchruns.tsp');
```

MCDG2LRA5yrb.m

```
%Monte Carlo Data Generator part 2 modified for rplotcross-section

PooSraw = [];

for i = 1:mcagarray(1)
    errdata = errdataruns(i,:);

        fn = datafilename(errdata);
        sc1 = dlmread(fn.sc1,',');
        sc2 = dlmread(fn.sc2,',');
        sc3 = dlmread(fn.sc3,',');
        magsc = FormAnal(sc1,sc2,sc3);
        legrateanalyzer5yr
        currentrun = ['LRA_' md5(CurRunVar(errdata)) ' - '
num2str(amyers) ' - ' num2str(i)];
        disp(currentrun)

    data.pulsestart12 = pulsestart12;
    data.pulseend12 = pulseend12;
    data.pulsestart13 = pulsestart13;
    data.pulseend13 = pulseend13;
    data.pulsestart23 = pulsestart23;
    data.pulseend23 = pulseend23;
    data.spikes12 = spikes12;
    data.spikes13 = spikes13;
    data.spikes23 = spikes23;

    binarray12 = zeros(1,1827);
    binarray13 = zeros(1,1827);
    binarray23 = zeros(1,1827);
    for i = 1:length(data.pulsestart12)
        binarray12(data.pulsestart12(i):data.pulseend12(i)) = 1;
    end
    for i = 1:length(data.pulsestart13)
        binarray13(data.pulsestart13(i):data.pulseend13(i)) = 1;
    end
    for i = 1:length(data.pulsestart23)
        binarray23(data.pulsestart23(i):data.pulseend23(i)) = 1;
    end

    PooSraw = [PooSraw,sum((binarray12 | binarray13) | binarray23) /
1827];

end

PooSmean = mean(PooSraw);
```

```
PooSstd = std(PooSraw);
PooSmax = max(PooSraw);
PooSmin = min(PooSraw);

PooSdata = LRAPooSLoad5yrb(mcagarray);

if isempty(PooSdata)
    PooSdata.mean = PooSmean;
    PooSdata.std = PooSstd;
    PooSdata.max = PooSmax;
    PooSdata.min = PooSmin;
else
    PooSdata.mean = [PooSdata.mean,PooSmean];
    PooSdata.std = [PooSdata.std,PooSstd];
    PooSdata.max = [PooSdata.max,PooSmax];
    PooSdata.min = [PooSdata.min,PooSmin];
end
LRAPooSSave5yrb
```

LRAPooSLoad5yrb.m

```
function [data] = LRAPooSLoad(mcagarray)

dbdir = 'c:\\data\\LRAdb\\';

load([dbdir 'LRAPooS5yr.mat']);
nstr = ['n_' num2str(mcagarray(1))];
pstr = ['p_' num2str(mcagarray(2))];
vstr = ['v_' num2str(mcagarray(3))];

if isfield(LRAPooSData, nstr) & isfield(LRAPooSData.(nstr), pstr) &
isfield(LRAPooSData.(nstr).(pstr), vstr)
    data = LRAPooSData.(nstr).(pstr).(vstr);
else
    data = [];
end
```

LRAPooSSave5yrb.m

```
%LRAPooSSave

dbdir = 'c:\\data\\LRAdb\\';

load([dbdir 'LRAPooS5yr.mat']);
nstr = ['n_' num2str(mcagarray(1))];
pstr = ['p_' num2str(mcagarray(2))];
vstr = ['v_' num2str(mcagarray(3))];

LRAPooSData.(nstr).(pstr).(vstr) = PooSdata;
save([dbdir 'LRAPooS5yr.mat'],'LRAPooSData');
```

rplotcs.m

```
%rplotcs - creates the actual rainplotcross-section

dbdir = 'c:\\data\\LRAdb\\';
```

```
load([dbdir 'LRAPooS5yr.mat']);
varray = fieldnames(LRAPooSData.n_50.p_0);
z = [];
zraw = [];
v = [];
vraw = [];
zverge = [];
zbuff = [];

for j = 1:length(varray)
    vcurr = char(varray(j));
    vcurrnum = str2num(vcurr(3:length(vcurr)));
    z = [z, mean(LRAPooSData.n_50.p_0.(vcurr).mean)];
    zraw = [zraw,LRAPooSData.n_50.p_0.(vcurr).mean];
    vraw =
[vraw,vcurrnum.*ones(1,length(LRAPooSData.n_50.p_0.(vcurr).mean))];
    for k = 1:length(LRAPooSData.n_50.p_0.(vcurr).mean)
        zbuff = [zbuff;mean(LRAPooSData.n_50.p_0.(vcurr).mean(1:k))];
    end
%     zverge = [zverge,zbuff];
    zbuff = [];
    n = length(LRAPooSData.n_50.p_0.(vcurr).max);
    v = [v,vcurrnum];
end

plot(v,z,'-k',vraw,zraw,'xk')
xlabel('V Error (\mum/sec)')
ylabel('Mean Normalized Time Out of Spec')
title('Leg Rate Rainplot Cross Section (P=0)')
```